A Thesis

entitled

Distributed Machine Learning Based Intrusion Detection System

by

Ahmad Maroof Karimi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Masters of Science Degree in Electrical Engineering

Dr. Weiqing Sun, Committee Chair

Dr. Ahmad Javaid, Committee Co-Chair

Dr. Hong Wang, Committee Member

Dr. Amanda Bryant-Friedrich, Dean
College of Graduate Studies

The University of Toledo
August 2016

An Abstract of

Distributed Machine Learning Based Intrusion Detection System

by

Ahmad Maroof Karimi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Masters of Science Degree in Electrical Engineering

The University of Toledo
August 2016

The proliferation of smart devices and computers has led to a huge rise in internet traffic and network attacks that necessitate efficient network traffic monitoring. There have been many attempts to address this issue; however, agiler detecting solutions are needed. In this research work, we have developed a distributed machine learning based intrusion detection system; and we have shown, IDS developed by us can monitor the network in near real-time. IDS, similar to other security tools such as firewall and anti-virus, are intended to protect the security of communication system. To achieve secure real-time network, we utilize the Apache Spark framework, Hadoop Distributed File System and Netmap a line rate packet capturing tool. The IDS is broadly divided into two major parts. 1) packet processing module and 2) machine learning based classifier module. The packet processing primarily includes feature extraction method, and we have presented the experimental results of the same for TCP-based traffic; we have also build traffic classifier based on Spark MLlib machine learning algorithms. The classifier updates the anomaly detection rule regularly; so that the IDS could continue to detect attacks even after a long time. We have implemented multiple machine learning classifier and choose the one with the highest accuracy. The results are shown along with the understanding gained for future work. With an adequate number of nodes, the IDS we have built can be set-up in organizations and institutions with huge traffic.

To my parents and all my closed ones who stood by me in good and tough times.

Acknowledgments

Firstly, I would like to thank God for everything. I would also like to show my sincere gratitude to my advisor Dr. Weiqing Sun for his initial guidance and continuous support during my MS program.

Besides my advisor, I would like to thank my co-advisor, Dr. Javaid for his constant help and he is always available to discuss and provide insightful comments on my work throughout my program. I would like to express my appreciations to my thesis committee member Dr. Wang. I would also like to acknowledge our department chair, Prof. Alam for his constant help and guidance since the first day of my program.

I would like to take this opportunity to acknowledge my gratitude to my Graduate Assistant Supervisor Ms. Sunday Griffith. It was a pleasure working with her and also a wonderful learning experience.

My sincere appreciation and thanks to all my labmates and particularly my senior Mr. Quamar Niyaz for all his support and enlighting research interest in me. It was memorable two years, and I enjoyed a lot being a part of Advanced Computing Research Lab.

Last but not the least, I would like to thank my family and my friends: my mother, her pearls of wisdom are always important in keeping me focused towards my goal, and my dad for he is the pillar of my strength. My elder brother Dr. Masroor, for always standing by me and guiding me in my academics and life in general and thanks to my younger brothers and sisters for their support and love. Big thanks to my grandmothers, for their never-ending prayers and good wishes.

Contents

Abstract iii

Acknowledgments v

Contents vi

List of Tables viii

List of Figures ix

List of Abbreviations xi

1 Introduction 1

1.1 Background 1

1.2 Related Work 3

1.2.1 Distributed Approach in IDS 4

1.2.2 Single Machine IDS 7

1.3 Goals and Objective 7

1.4 Outline 8

2 Intrusion Detection System Architecture 9

2.1 Monitored Network 11

2.2 IDS 12

2.2.1 Traffic Collector and Monitor 12

2.2.1.1 Netmap 12

2.2.2 Spark Cluster 14
2.2.2.1 Hadoop HDFS 14
2.2.2.2 Apache Spark 15
2.2.3 Machine Learning Module 16
2.2.3.1 Naive-Bayes 18
2.2.4 Support Vector Machines 20
2.2.4.1 Decision-Tree 20
2.2.4.2 Random Forest 21

3 Network Traffic Packet Feature **23**
3.1 TCP/IP Packet Sniffing 24
3.1.1 Transmission Control Protocol and Internet Protocol: 25
3.1.1.1 TCP Segment 27
3.2 Packet Processor 32

4 Traffic Classification **38**
4.1 Traffic Dataset 38
4.1.1 Attack Dataset 38
4.1.1.1 Types of TCP attacks 39
4.1.1.2 Attack Tools 41
4.1.1.3 Attack Testbed and data generation and collection . 42
4.1.2 Normal Dataset 43
4.2 Traffic Dataset Analysis 43

5 Conclusion and Future Work **52**
5.1 Summary 52
5.2 Future Work 53

References **54**

List of Tables

3.1 Traffic features extracted for TCP DDoS attack detection [20] 36

4.1 Analysis of network traffic using Naive-Bayes algorithm 47

4.2 Confusion Matrix 49

List of Figures

1-1 A simple diagram showing Network Based Intrusion Detection System (NIDS). 3

2-1 Architectural diagram of the developed IDS. The IDS consits of two major components: i) Traffic Collector and Monitoring System(TCM) and ii) Spark Cluster System 10

2-2 Flow chart showing various steps involved in Intrusion Detection System. 11

2-3 An comparision between dumpcap, tshark and netmap-libpcap under TCP flooding using hping3 tool for network traffic collection. 13

2-4 Flow diagram of various stages of supervised machine learning 17

2-5 Information Gain curve with the variation in fraction of sample in complete classsize 21

3-1 TCP/IP packet stack [1]. 25

3-2 Three-way handshake connection process between two machines [1]. . . 26

3-3 Diagram showinng fields of TCP packet header. 29

3-4 Diagram showinng fields of IP packet header. 30

3-5 screenshot of csv files showing captured header fields. 31

3-6 Traffic distribution during network intrusion, analyzed from CAIDA dataset [2]. 33

3-7 Time taken by varying file size to process with change in number of nodes. 34

3-8 Snapshot of the features calculated by spark cluster. 37

4-1 Figure showing the amplification attack setup. 39
4-2 Figure showing the testbed setup to generate attack dataset. 43
4-3 Figure showing the normal traffic pattern. 44
4-4 Figure showing the anomalous traffic pattern. 45
4-5 Figure showing the normal traffic pattern. 46
4-6 Receiver Operating Characteristic(ROC)from the SVM algorithm. 48
4-7 One of the decision tree from a random forest algorithm. 49
4-8 Graph showing the percentage accuracy of three classifier. 50
4-9 Graph comparing the precision, recall and F-measure of three classifier. . 51

List of Abbreviations

DDoS	Distributed Denial of Service
FPR	False Positive Rate
HDFS	Hadoop Distribute File System
IDS	Intrusion Detection System
MLlib	Machine Learning Library
RDBMS	Relational Database Managment System
RDD	Resilient Distributed Dataset
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TCM	Traffic Collector and Monitor
TPR	True Positive Rate
YARN	Yet Another Resource Negotiator

Chapter 1

Introduction

1.1 Background

The profusion of the e-devices and extensive reliance on web based applications for both, our regular activities, as well as high-profile use has led to ever increasing network/internet communication. This has led to the generation of a tremendous amount of traffic data at a very fast rate, posing some serious challenges for safe and reliable use of the internet by individuals and institutions. As per the available data, over the years there has been a tremendous increase in network traffic and a corresponding rise in network intrusion or cyber attacks. Based on the Cisco reports, the size of the global internet traffic will reach zettabytes (10^{21}) by the year 2016 and twice by the end of the year 2019 [3].

Cyber attack or network intrusion is an activity which tries to compromise the normal functioning of a computer network. To neutralize cyber attacks, we have to develop a mechanism called intrusion detection, which is a method to mitigate or report these intrusions. However, it becomes difficult to monitor and identify intrusions at very high network speed and moreover, in the event of an outbreak of Distributed Denial of Service (DDoS) attacks, these issues aggravate exponentially. Therefore, it becomes imperative on the part of organizations to equip themselves against immi-

nent network attacks. With legacy intrusion detection methods, we have struggled to keep a watch on the networks efficiently. To overcome these challenges, in the recent years, there have been various attempts to propose efficient Intrusion Detection System (IDS). IDS is an application that monitors, detects, and prevents the network or the system against any suspicious activity of harming network's Confidentiality, Integrity, and Availability (CIA) properties. It includes monitoring of unwanted utilization of the network resources, keeping it available for the legitimate users and in some cases preventing loss of information/data to the intruder. IDS can be classified into several ways and some of the classifications are:

1. Network-based and Host-based [4]:

   In Network-based Intrusion Detection System (NIDS), each packet/flow in the network, whether inbound or outbound is monitored by sniffing all the packets coming at the interface of IDS. IDSs are installed strategically at multiple locations in the network like Backbone/Core layer, Distribution layer, and Access layer. Whereas, for Host-based Intrusion Detection System (HIDS) setup, IDS programs instead of a network are installed in every system of the network.

2. Passive and Active [5]:

   An IDS, which only raises an alarm in the event of an intrusion, is known as passive IDS. An IDS, which also takes action in addition to raising an alarm in response to intrusions, is called active IDS.

3. Signature based and Anomaly based [6]:

   Signature based IDS is based on matching with a pattern of known attacks pulled from a data file with activities having malign influence. Anomaly based IDS identifies the anomalies in the traffic, and it can detect unknown/new attacks in the network because it relies on the rules as opposed to signature based IDS.

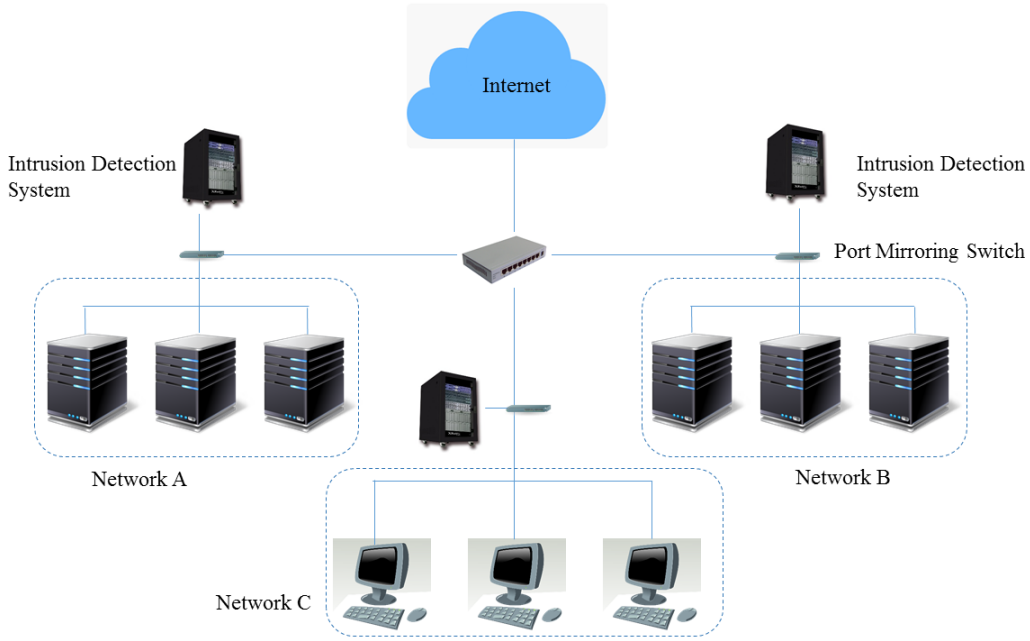

Figure 1-1: A simple diagram showing Network Based Intrusion Detection System (NIDS).

We have narrowed our research focus on anomaly/statistical based NIDS. Anomaly based approach uses a statistical method to learn the behavior of features and then to identify the attack, it looks for deviation from the normal behavior. The fundamental principal involved behind any IDS is to monitor all the data coming to the switch's interface. This approach requires, first capturing all the packets that pass through observed switch and then analyzing these packets to detect aberration in behavior. In figure 1-1, we have shown the abstract diagram of the IDS.

1.2 Related Work

There have been many endeavors in the area of network traffic analysis. Most of these efforts incorporate the use of single machines with the significant limitations in storage space and computational resources. In this section, we will mainly explore

the works which discuss the use of distributed computing for network monitoring. After all, the essence of our research is imbibed in distributed computing. At the same time, we will also mention briefly, the approach that is not distributed but can be equally efficient in some cases.

1.2.1 Distributed Approach in IDS

One of the first few works in this field is done by [7]. They proposed and developed a Hadoop-based MapReduced Parallel Packet Processor tool to process network packets. The key points in their work are 1) Writing packet trace files on Hadoop cluster in HDFS. 2) Compared to traditional techniques, Hadoop-based MapReduce programs are much more computationally efficient. 3) Developed Binary input/output PcapInputFormat module. 4) Designed MapReduce model considering both Mapper and Reducer tasks. They tested their work on four and ten nodes Hadoop cluster and compared the performance of these two with CAIDA real-time network monitoring application, CoralReef. The paper mentions that four node cluster isn't much better than CoralReef. The reasons quoted are reading and writing data on disk I/O cost enough time, hence neutralize the combined performance of four machines. However, when ten nodes high-performance devices compare with CoralReef, overall cluster performs ≈ 7 times faster. Since, they have employed Hadoop, so their system automatically provides fault tolerance. Albeit this was innovative project, but over the years, with the advent of new frameworks and tools, there's a need to improve further, the processing capability of this system [8].

The authors of [9] have also developed the scalable system to detect peer-to-peer Botnet attacks using machine learning technique. They have used Hadoop and some of the tools from its eco-system like Hive, Apache Mahout. They have claimed their system to be quasi-real-time, a step towards the fully real-time system. Their application can be broadly classified into two stages. 1) Sniffing network traces and then

processing these packets to extract features using Hive. 2) Machine learning based model to detect Botnet attacks. In their work, they have used several other tools like Dumpcap and Tshark. Packets are first saved on the disk in pcap format and then delimited files are stored on HDFS. Thereupon Hive query language is used to extract relevant features for machine learning. To get the network attack dataset, they built test-bed to generate botnet attacks by installing KelihosHlux, Zeus, and Waledac on appropriate computers. This system works, fine with a latency of $\approx 30s$, but it also has its limitations regarding low network bandwidth and high packet loss.

In [10], the authors have worked on developing a system based on Deep Packet Inspection for an e-commerce website to identify the products from digital code. Although, this work is not related to monitoring attacks in the network but the challenges faced by them are pretty similar to ours. The primary challenge, they encountered was in handling the enormous amount of web traffic. To address this problem, they come up with Hadoop-based Packet Analyzer. Overall their system is broadly classified into two categories: 1) Web Crawler and 2) Hadoop. In the first part, the main function is to capture the packets and parse it to extract relevant information and the store it in the HDFS. In the second part, MapReduce program is used as an Analyzer to identify the product. To test their system, they had setup a total of 12 nodes VM cluster on four different machines. First, they compared processing time by varying the input file size and found that file size equal to block size are ≈ 5 times faster than huge file sizes. Later, they analyzed 310 GB of files.

In [11], the authors have proposed a Hadoop MapReduce framework for anomalies detection in the network. In this work, they developed Hashdoop framework to split the data in HDFS, so that network packets can maintain their structure. The overall process is distributed into two parts, in the first part, traffic is split using hash function then a MapReduce function identifies the anomalies in each split and consequently raises an alarm. To evaluate their model, they setup the cluster of 6 nodes and the

dataset from MAWI archive [12]. They successfully gain the speedup of 3.5 times faster than a standalone system.

In [13], Tazaki et al. have discussed three important properties of DDoS detection framework. These properties are scalability, real-time analysis, and uniform programmability. They developed a Hadoop-based platform called Matatabi. This framework has four components: 1) Scalable HDFS to store data and provides an abstraction to the various utilities running on top of HDFS, e.g. tools like HIVE and PIG, which are part of Hadoop eco-system can run on top of HDFS. 2) Data Import Module provide locality of data to MapReduce applications running on top of Hadoop. 3) Analysis Module, consist of MapReduce programs or other distributed computational tools like HIVE, etc. 4) MATATAPI, an API for accessing the results after analysis. The system currently runs every 24 hours, and they are planning to improve it frequency and also real-time in some cases.

Authors of [14] have developed a tool to monitor network traffic called Tstat, which implements network traffic classifiers. It is a highly flexible tool and can be configured even at run time. Tstat applies three packet inspection techniques. First, application signatures are collected, and pattern matching approach is used to check the payload against all signatures. In the second case, inspection involves matching set of specific rules for a flow. The third case, requires investigation of statistical properties of the packets.

In [15], the authors have described the detection of abnormal behavior in the event of cyber-targeted attack based on big -data processing and storing technique. This work focuses on detecting Advance Persistent Attack before the final attack can be executed. The system detects abnormal behavior by analyzing the association between response to an event of a security equipment and statistical information. The application is divided into multiple layers, beginning with data collection layer. This layer receives data from data sensors and saves it on Hadoop HDFS. Next layer is data

processing layer; it consists of HBase and MySQL cluster. MapReduce program and Apache Storm is used as processing tools. The third layer is targeted attack analysis layer and is composed of network-based, host-based and legacy Security Analysis engine. The final layer is integration layer; it comprises of visualization engine to provide results of analysis layer. The test environment consists of 12 Nodes machine with the real-time capacity of 64GB and after expansion, it will be 192GB with data storage of size 55TB.

1.2.2 Single Machine IDS

Arian Bär and et al. in [16] have proposed DBStream for network monitoring against intrusion attack. Even though it is not a distributed system but, we have included it here because of its high-performance capabilities when compares to several nodes of distributed system. DBStream is an SQL based rolling data analysis tool. The two major points addressed by this work are: 1) Support of incremental queries, which means, queries that can run on the new dataset and combine the output of previously calculated data. 2) The design of DBStream is such that it utilizes DBMS query processing engine and the query, optimizer. They have also given the performance comparison analysis with Spark batch processing, and in some cases, DBStream gives results comparable to 10 nodes cluster.

1.3 Goals and Objective

It is well understood that nowadays network security is indispensable for any organization or institution and hence the need for IDS. Although, there have been quite a few research works in the field of distributed computing traffic monitoring. There has always been the need to improve processing capabilities of these systems further and increase the storage capacity to handle to compete with ever growing network traffic.

Almost, all of the research work we have gone through on distributed environment, have implemented Hadoop-based systems to detect threats in the network. However, Hadoop's multiple reads and write actions on the disk creates a huge bottleneck in the performance.

The focus of this work is to overcome the challenges in the previous work. The major point addressed by us are:

1. Enhancement in the computational performance of IDS by using latest tools and frameworks. The main points of our application are the traffic packets collection, extraction of traffic features using Apache Spark framework and storage of network packets and features on the HDFS.

2. Build classifiers based on Apache Spark machine learning libraries to update the anomaly detection rule regularly.

1.4 Outline

This section is composed of the organization of the rest of the chapters. In chapter two, Intrusion Detection System architecture is described. It gives the graphical illustration of the IDS architecture and the figure also depicts the flow of data through different stages of the IDS.We will also provide the brief introduction to the various tools we have used in developing our IDS. In chapter three, we have described the various features we have collected of the network traffic packets. We have also described the approach we have taken to collect those features. This chapter, also comprised of the techniques we have used to process and store packet features in distributed system. Chapter four discusses various machine learning techniques used to develop a binary classification model. We have also mentioned about the dataset collected to train our model. In the last chapter, we have presented concluding statements of our work and the future work we are looking to do on this project.

Chapter 2

Intrusion Detection System Architecture

We have proposed and developed an IDS to detect and avert DDoS intrusions in the network in real-time. In figure 2-1, we have shown the architectural diagram of the IDS system we have developed. The figure is divided into two parts: 1) Monitoring system called IDS and 2) Networks to be monitored. Port mirroring switch act as an interface and connects the two parts.

IDSs can be placed at different strategic locations in the network; installing it is a matter of mutual concession and compromise because, if we install the IDS at a place with large network traffic flows, it will lead to huge computational cost and may be increased latency. However, if we install IDS in a place with very limited traffic, then the number of IDS to be installed have to be increased to monitor the complete network. So, IDS have to be placed carefully and judiciously in the network, so that computing resources are utilized optimally and at the same time there is no or low latency in response while monitoring a critical system.

In our System, all the networks to be monitored are connected to the port mirroring switch. The port mirroring switch replicates the flow of traffic of all the switches at one single switch. The replication of traffic at one switch helps us in collecting

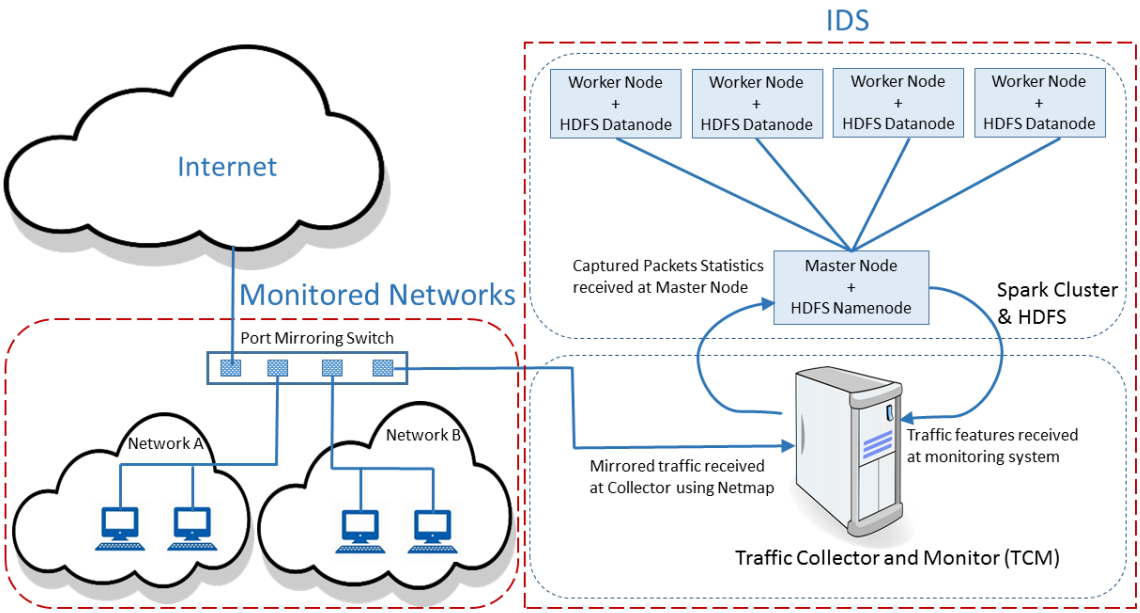

Figure 2-1: Architectural diagram of the developed IDS. The IDS consits of two major components: i) Traffic Collector and Monitoring System(TCM) and ii) Spark Cluster System

the traffic data of all the networks connected to the port mirroring switch. As shown in figure 2-1, each network has multiple machines and these machines communicate among themselves and also communicate across the network, but since we are only monitoring internet traffic, we have not installed port mirror switches in intra-network communications.

In figure 2-2, we have shown the flow chart of the various steps involved in intrusion detection. It illustrates that extracted features of the network traffic are passed through classifier model to examine the traffic, and a copy of the data is moved to HDFS for training future classifier models. This activity prevents the IDS from becoming outdated over an extended period. In next section, we will describe both components of the system architecture.

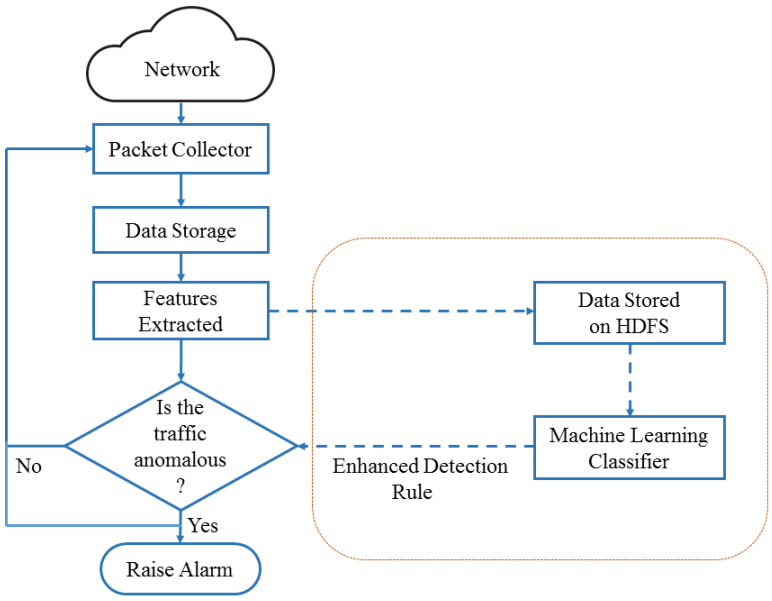

Figure 2-2: Flow chart showing various steps involved in Intrusion Detection System.

2.1 Monitored Network

Monitored networks are the networks with a set of computer machines which allow them to communicate and transfer data. The communication can be internal and as well as over the internet. Our focus is on the internet/external traffic over the network. We have placed port mirroring switch such that whatever packets comes to the interface or flows out of the network are replicated. We have collected data from our school department network, and we have also collected data over the home network with around ten machines. In both these networks, we have made one of the switches as a port mirroring switch and connected a traffic collector to it as shown in the figure 2-1.

2.2 IDS

IDS has two major monitoring components. They are 1) Traffic Collector and Monitor, and 2) Spark Cluster and HDFS. We have explained them in section 2.2.1. Spark cluster and HDFS is further sub-divided into packet processing module and machine learning module.

2.2.1 Traffic Collector and Monitor

Traffic Collector and Monitor (TCM) is a server to live capture and store packets header in a file intermittently before transferring the accumulated files on HDFS. We have installed Netmap-libpcap tool in the server for collection of packets. Sniffing or capturing network packets at a line rate is a prerequisite for building an IDS, which monitor networks at real-time.

2.2.1.1 Netmap

Netmap-libpcap is a tool that provides us very high-speed packet sniffing capability [17]. A 10 Gigabit/s NICs can handle 14 million packets per second and the frequency can speed-up to reach 30 million packets per second at 40 Gigabit/s NICs. Since Netmap is hardware independent, its implementation is easily compatible with Ubuntu machines we are working on, with little modifications. The Netmap-libpcap provides high speed by overcoming the three major bottlenecks in other traditional packet sniffing applications, and they are: 1) Memory Allocation, Netmap provides preallocated memory instead of dynamic memory allocation. 2) Reduced system call overheads, instead of one call per packet it supports system calls over large batches, resulting fewer calls for entire data. 3) Zero-copy transfer of packets between various interfaces. After sending the captured packets to Spark cluster for monitoring it expects to receive the processed traffic features back from the cluster. Once the packet

features are received, monitoring aspect of TCM plays its role. In TCM server, we have also deployed machine learning based analyzer models to detect intrusions.

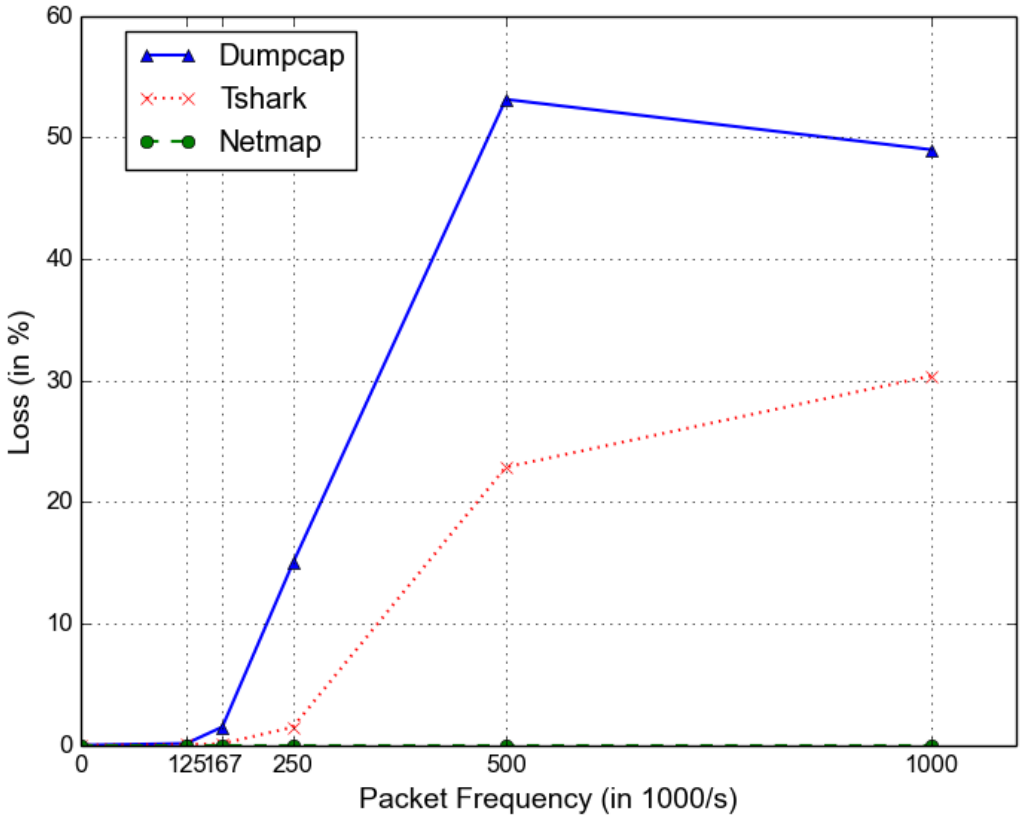

Figure 2-3: An comparision between dumpcap, tshark and netmap-libpcap under TCP flooding using hping3 tool for network traffic collection.

Before selecting Netmap-libpcap for fast packet sniffing tool, we compared netmap with Dumpcap and Tshark. These tools are used in popular existing IDS for packet sniffing, and they have been used extensively in previous related works we discussed in Chapter 1. We run an experiment for comparing packet capturing capability of all these tools. We used hping3 [18], a tool to create flooding in the network, by dispatching packets at very fast pace from one machine to host machine. In our experiment, we changed the frequencies of packet in the network from 0.125 million packets per second (Mps) upto 1 Mps. Figure 2-3 shows that when we increase the frequency of packets flow in the network, packet loss observed is quite high in case of

Tshark and Dumpcap. However, it is negligible in the case of Netmap-libpcap even at very high frequency. The important point to be noticed is that, unlike other tools, with Netmap-libpcap, in addition to capturing packets we were also able to extract header fields with Netmap-libpcap with a much better performance. It prompted us to use Netmap-libcap for packet capturing.

2.2.2 Spark Cluster

Spark Cluster (SC) is a set of machines for running Spark jobs on the top of Hadoop HDFS. So, we have made all of the Spark worker nodes as a Hadoop datanode for storing files with the packet header. These files act as an input to the scala application deployed on Spark cluster for packet processing. The brief introduction to Hadoop HDFS and Apache Spark is presented in following sections.

2.2.2.1 Hadoop HDFS

Hadoop is a distributed computing framework for data storage and processing. The distributed storage space provided by Hadoop is called HDFS. One of the basic objective of HDFS in our work is to provide scalable and reliable disk space for storing a large volume of captured packet header data. HDFS achieves high reliability on account of replication of data on different nodes. The total number of times the data stored on HDFS can be replicated is configured during the initial setup of Hadoop, It can be changed by updating the configuration file and we have kept it to three. Even though Hadoop is developed to run on commodity hardware, combined resources of multiple machines in a cluster gives HDFS the ability to provide high throughput to the application [19]. Along with providing scalability of disk space, HDFS provides an abstraction of data to Spark applications and developer need not care about the distribution of blocks of data on different nodes of HDFS. Spark application connects to Hadoop either through standalone mode or Hadoops YARN manager [20]. In the

standalone mode, job scheduling is done by the spark, and in the Hadoop's YARN mode by Hadoops resource manager. Spark and Hadoop clusters are easily scalable, and we have varied the number of nodes from four upto twelve.

2.2.2.2 Apache Spark

Applications involving high velocity and a huge volume of data needs fast processing requirements and large storage space. Although, distributed system like Hadoop is great in providing scalable distributed storage space. However, it lacks efficient data processing capability because of its inherent processing method of multiple disk I/O operations. We have included Apache Spark particularly to address this issue [21]. Apache Spark is an open-source, cluster computing framework for fast processing of data. Spark provides the feature to hold in-memory data that are generated due to various stages in data transformations. Spark performs upto 100 times and 10 times faster processing than Hadoop for cached/in-memory and disk data, respectively [22]. One of the important concepts of Spark is that it provides an abstraction on the collection of objects called Resilient Distributed Dataset (RDD) which are partitioned across worker nodes and can be operated upon parallelly. The other significant property of the RDD is, they are fault-tolerant and are stored in the form of read-only immutable RDDs. Spark can cache them in memory for computations. Although RDDs are immutable, but various functions available in Spark like map, reduce, flatMap, filter, take, collect, and count [22] transform the RDDs from one state to another. Spark also provides useful built-in libraries, and we have used some of them like Spark SQL, MLlib, and Dataframe. These APIs are developed on the top of spark to provide a system that gives abstraction on the datasets we want to work. Spark SQL provides the relational operations on datasets pretty similar to relational databases. It runs on top of Spark framework and converts an RDD to a dataframe, which is equivalent to a table in RDBMS. The Spark dataframe provides the users a

platform to interact with data using Spark SQL.

2.2.3 Machine Learning Module

Machine Learning is a technique to train a model by providing the ample amount of past data to predict future patterns. So, if the model learns the pattern successfully, there are high chances it will predict correctly. Machine learning techniques are commonly applied to the problems which can not be solved by writing code or by mathematical means alone. Machine learning problems have two distinct approaches, and they are classified as Supervised Learning and Unsupervised Learning. Since we have used supervised learning, we will explain it further in brief.

In supervised machine learning, algorithms are trained by providing them with pre-define collected data. This data is first labeled and based on these tags, the algorithm learns and then develops a model. The developed model then facilitates the accurate result when given a new data. The learning depends on the sophisticated algorithms and the training data. To give an idea about the working of training algorithm, we explain one of the simplest equation of the predictor function, where the predictor function or hypothesis function depends on two input values. It can be written in mathematical expression as follows:

$$h(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2$$

Where Θ_0, Θ_1 and Θ_2 are coefficients and x_1 and x_2 is a independent variable or input data. General form of this equation can be written as follows:

$$h(x) = \sum_{i=0}^{n} \theta_i x_i$$

Where n is the total number of input data features or total number of input variables. When we train an algorithm, we actually find the values of coefficients. So if new data is passed to the function, based on the values of the input features, predictor

function calculates the value. Depending on the value, whether it is below or above the threshold, new data is designated its class. We have shown the steps of supervised machine learning in figure 2-4.

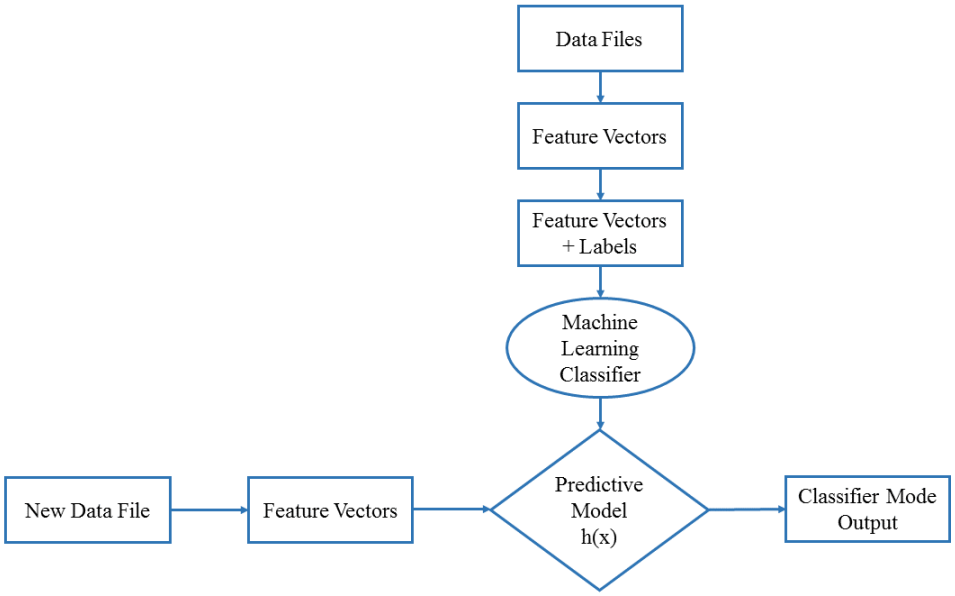

Figure 2-4: Flow diagram of various stages of supervised machine learning

Further, under supervised machine learning, there are two important categories: 1) Regression and 2) Classification techniques. In regression method, the output of the hypothesis function lies on continuous spectrum whereas, in Classification technique, the output of the prediction function is in the form of distinct discrete classes.

We have incorporated the machine learning module in our application to enhance periodically the anomaly detection rule deployed in TCM. We have used Apache Spark scalable machine learning library MLlib [23] for our purpose because of many of its advantages. It has built-in classification based algorithms like Decision Tree, Random Forest, and Naive-Bayes. MLlib has spark.ml higher-level API built on top of Spark dataframe, and it gives more versatility and is easy to use [23]. Just like other Spark libraries, MLlib can run on Hadoop HDFS and because of its in-memory

feature, Spark MLlib are quite fast.

As mentioned earlier, packets features calculated in Spark Cluster is sent over to the monitoring system to detect any attack in the network, but at the same time, we store the collected features in HDFS for over a longer term. Since the pattern in the network traffic does not remain same and changes over the period, and hence there arises the need to update the trained model of a monitoring system in TCM. So whenever, there is a need to re-generate intrusion detection model or hypothesis function, we have enough data available to train.

In our work, the three machine learning algorithms we have applied are Naive-Bayes, Decision Tree, and Random Forest. For training the model and then testing its accuracy, precision, and other parameters, we used the 70 : 30 ratio of test and training dataset. In the following, subsections, we will briefly describe the algorithm employed.

2.2.3.1 Naive-Bayes

It is a simple multiple-class classification algorithm with an assumption that a value of every feature is independent of the other regardless of any correlation between different features [24]. Despite being simple, Naive-Bayes algorithm has been quite successful in many physical world problems. It is based on Bayes' theorem, and it first calculates the conditional probability of each feature for a given label and then applies Bayes theorem to get the probability. Bayes theorem provides a method to find the posterior probability $P(c|x)$, given $P(c)$, $P(x)$, and $P(x|c)$ by following rule:

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)}$$

where :

$P(c|x)$ it gives posterior probability for a class for given features x.

$P(c)$ is the prior probability of class.

$P(x|c)$ is the prior probability of features, given class.

$P(x)$ is prior probability of features.

But in our case $P(x)$ is not known, so there is another way round to solve this equation:

$$P(c|x) = P(x_1|c) * P(x_1|c) * ...P(x_n|c) * P(c)$$

Here, frequency table of each feature is drawn against the target and multiplied with the probability of class. Once, we get the posterior probability of each class; we compare them, and the class with higher probability gets labeled for particular data. It is a simple multiple-class classification algorithm with an assumption that a value of every feature is independent of the other regardless of any correlation between different features [24]. Despite being simple, Naive-Bayes algorithm has been quite successful in many physical world problems. It is based on Bayes' theorem, and it first calculates the conditional probability of each feature for a given label and then applies Bayes theorem to get the probability. Bayes theorem provides a method to find the posterior probability $P(c|x)$, given $P(c)$, $P(x)$, and $P(x|c)$ by following rule:

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)}$$

where :

$P(c|x)$ it gives posterior probability for a class for given features x.

$P(c)$ is the prior probability of class.

$P(x|c)$ is the prior probability of features, given class.

$P(x)$ is prior probability of features.

But in our case $P(x)$ is not known, so there is another way round to solve this equation:

$$P(c|x) = P(x_1|c) * P(x_1|c) * ...P(x_n|c) * P(c)$$

Here, frequency table of each feature is drawn against the target and multiplied with the probability of class. Once, we get the posterior probability of each class; we compare them, and the class with higher probability gets labeled for particular data.

2.2.4 Support Vector Machines

Support Vector Machine (SVM)[25] is a classification technique based on the concept of decision planes or hyperplanes that define class boundaries. A decision plane divides set of objects having different class into different categories. For a simple case, decision plane can be a straight line. However, for complex scenarios, a set of mathematical functions called kernels are applied to divide the data across a straight plane in a feature space of a higher order. New data points are then mapped to any one of the classes depending on the category they belong.

2.2.4.1 Decision-Tree

Decision Tree builds classification models by splitting the training dataset based on values of the selected features. Features are divided across a value in a recursive manner, breaking the dataset into smaller subsets and in turn generating a tree called Decision Tree. Nodes in between the tree are called decision nodes; the terminal nodes are known as leaf nodes, and the topmost node is called root node. The algorithm to build decision tree is called ID3 and ID3 uses Entropy and Information Gain. Entropy is known as the measure of uncertainty.

$$E(S) = \sum -p_i log_2 p_i$$

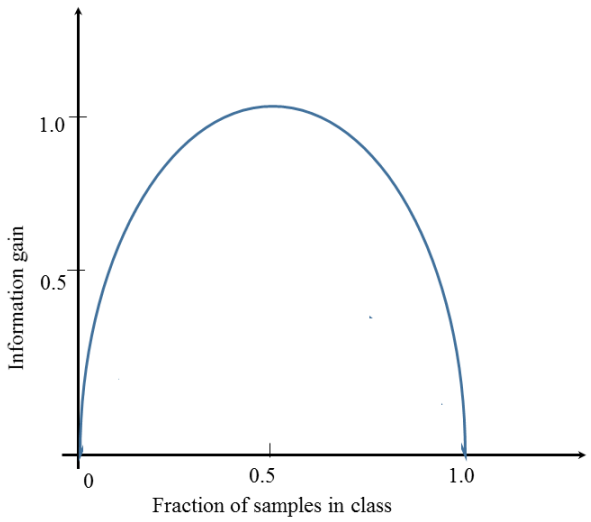

Figure 2-5: Information Gain curve with the variation in fraction of sample in complete classsize

$where : p_i$ is the probability of the class.

The amount of Information gain or decrease in the entropy once data is split, indicates importance of an attribute in determining the target.

$$Information\ Gain = entropy(parent\ node) - entropy(child\ node)$$

In other words, we can say from the figure 2-5, when the fraction of samples in the data is exactly half, the information gain in selecting that node is maximum. On the other hand, when a sample is homogeneous, and there is only one kind of sample in data, Information gain is minimum because entropy is already minimum.

2.2.4.2 Random Forest

Random Forest is an algorithm of an ensemble of multiple Decision Trees. Each tree of a set gives its prediction result, and the label is given to the class with the most number of prediction in its favor. One of the main advantages of random forest

is that it reduces the risk of over-fitting. More the number of trees in the random forest it's easier to tune its prediction. Randomness in a random forest is generated by selecting a random subset of training data to build trees. Subset data can have all the input attributes but a limited number of records or all the data records but a subset of attributes or subset of records and attribute both. Random Forest takes care of an unbiased estimate of the test set error, and it is determined internally during run-time. [26].

Chapter 3

Network Traffic Packet Feature

To analyze a network traffic, it becomes crucial to understand the communication over a network. The basic building block of communication is network packets. The network packet is a basic unit of information or buffer containing data, that is routed from sender machine to receiver device over the internet or other networks. Apart from the actual data which is called payload, packets also contain other parts called a header and a trailer. The TCP/IP network model has a four-layer and each layer of the model, as shown in figure 3-1 builds the part of the packet before sending it to the receiver. The packet switching technology used in routing the network packets helps them reach the destination through multiple numbers of transmission points/hops in the network making it free from having a dedicated route from a host to the client machine. Now, we will present brief description explaining different parts of the packet.

Payload: A payload/data is a primary data that a packet has to deliver to the destination. Normally packet is of variable length, but if a packet size if fixed in length, then the data to be sent is padded with empty bits to make it off the right size.

Trailer: Trailer is the portion of a packet that carries the information indicating the end of the packet. It also provides the feature for checking a violation in the integrity property of packet.

Header: A header is a part of a packet that contains all the necessary information required to send it to it's intended destination. It is a set of a structured field containing information of source and destination. Since each layer of the network has some rules, the IP packet is built in parts over as shown in figure 3-1 Each layer, as the data moves from application layer to data link layer adds its header on the data of the higher layer. The network header is a combination of the header of the all four layers, and they are described below.

3.1 TCP/IP Packet Sniffing

One of the first steps in building an IDS is to sniff packets from the network. As mentioned in section 2.1, to accomplish this task, we have installed Netmap-libpcap tool in our Traffic collector module. While sniffing the packets from the network, our focus is only on the packet headers, leaving its payload and trailer.

According to [3], about 75% of the global network attacks in the range of 50-100 Gbps occurred in North America. Out of these, almost 99% of the attack is TCP/IP attack in Q2 2015 [27]. Keeping these statistics in mind, we have focused our work on TCP/IP attacks. So, we have designed this IDS only for TCP/IP packets, while it can easily be extended to other protocols. Before proceeding further, it is useful to understand the TCP/IP packet communication method. It will help us understand vulnerabilities and threats associated with it. In figure 3-1, it is shown how the exact packet is formed in TCP/IP protocol stack. Here, we have also given a short description of TCP/IP, explained the subtleties in establishing its connection and header fields of TCP and IP segment.

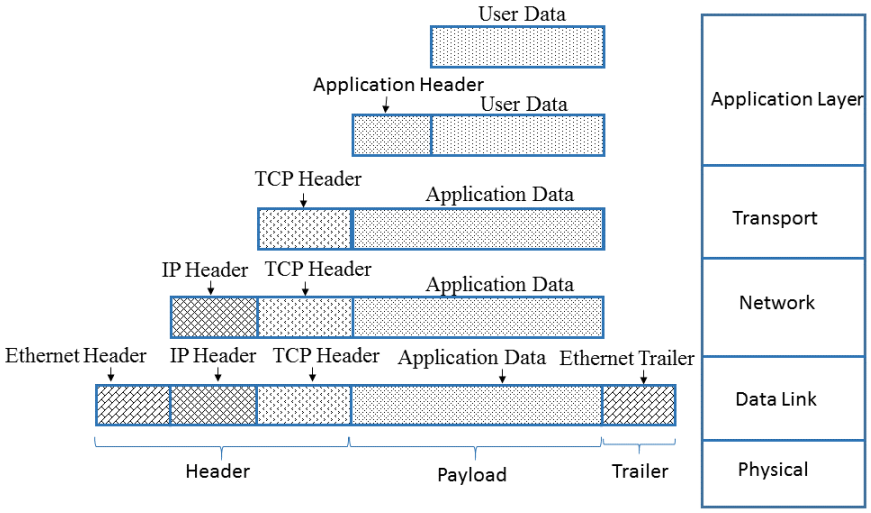

Figure 3-1: TCP/IP packet stack [1].

3.1.1 Transmission Control Protocol and Internet Protocol:

TCP/IP is a set of TCP and IP protocols to establish a connection between two machines for communication. Internet Protocol (IP), guides the packet from host to the client machine and TCP protocol ensure the packet is delivered in order and reliably. In the case of any error, it resends the packets. TCP transport layer protocol is connection oriented, which means that before formal communication begins between client and server, a connection needs to be established [28]. This establishment of a connection is popularly known as Three-way handshake. Attackers have exploited TCP/IP communication channel to launch network attacks. Establishment of connection is done as follows:

- First, client node sends SYN packet to the server to check whether; the server is open for connection. In the process, client increments its sequence number

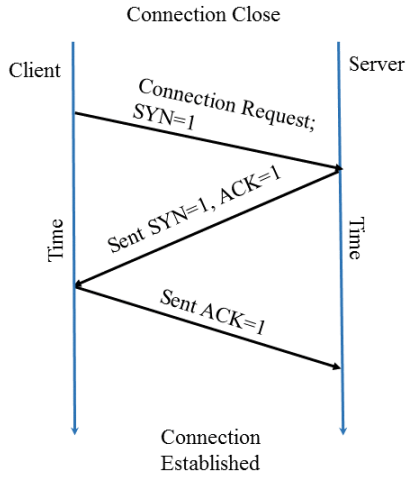

Figure 3-2: Three-way handshake connection process between two machines [1].

field by one.

- Second, if the server is ready to open a new connection, transport layer of the server will send a packet to the client with SYN and ACK flag set. Signaling acknowledgment, the server sends acknowledgment number by incrementing 1 to the sequence number received from the client.
- In the third step, when the client receives the packet with SYN and ACK flag set, it replies back by sending a packet to the server with ACK flag set.

Once these three steps are completed successfully, we say TCP connection is established, and client-server sequence and acknowledgment number get synchronized, and now the client and the server can communicate. In figure 3-2, we illustrate Three-way handshake connection.

Now, as we know the process of creating a connection, there's also a way to terminate the connection. It's a symmetric, independent process for both client and

server to close the connection from their end. From the client's end the process is following:

- First, the client sends a packet to the server with FIN and ACK flag set. FIN flag indicate, there's no more data to be sent, and ACK flag identifies the particular connection they have established and now want to close.
- Second, the server sends a packet acknowledging the FIN received from the client.

To close the server's connection, above two steps should be repeated from the server's end.

The other important step in designing a IDS is to understand the header fields of the TCP and IP segments. Figure 3-3 gives the graphical representation of the TCP header and figure 3-4 represents the header fields of IP header. We will discuss them briefly here:

3.1.1.1 TCP Segment

- Source Port: 16 bits port number for the process that initialized the communication and this port is for a reply from the server machine or request sent from the client to server.
- Destination Port: 16 bits port number, this is the final destination of the message sent by the client and also for a reply from the server.
- Sequence Number: Its used in two-way, initially at the time of establishing the connection to synchronize client and server and afterward to maintain the bytes of data sent.
- Acknowledgment Number: This field acts as a response, and it contains the sequence number the source is expecting from the destination.

- Data Offset: It signifies the offset of the data from the first bit of TCP segment because of the header size.
- Flag fields: They are 6 bits, and are also called control bits.
  - URG: This is urgent pointer bit when set, it causes the receiving machine to forward the critical data on a separate channel. It allows the application to process the data out of band breaking FIFO rule followed in normal operation.
  - ACK: Acknowledgment bit when set, identifies that the TCP segment is carrying an acknowledgment.
  - PSH: If this bit is set, it recognizes the request that data be sent to be pushed immediately.
  - SYN: This bit is one to establish a new connection and synchronize the sequence number.
  - SYN: This bit is set to establish a new connection and synchronize the sequence number.
  - FIN: When finish bit is set to 1, it indicates sender wants to close the connection.
- Window field: Size of the sender and receiver window, 16 bit. It shows the amount of data the sender of this packet can accept from the receiver side at a time and the size of the sending segment of the receiver.
- Checksum: 16 bits field for checksum and it ensures the integrity of the data.
- Urgent Pointer: This field of 16 bits contain the sequence number of the last byte of urgent data, and it's used in conjunction with URG flag.
- Reserved: 6 bits always sent as 0.

IP Segment

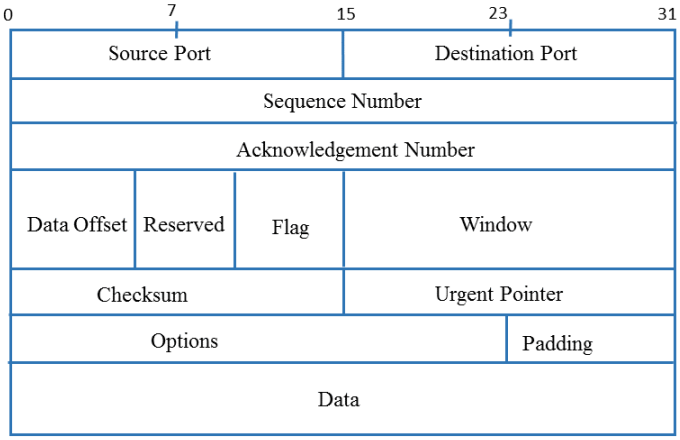

Figure 3-3: Diagram showinng fields of TCP packet header.

- Version: This field identifies the version of the packet being sent. It can be either for IPV4 or IPV6.
- Internet Header Length (IHL): It gives the length of the IP packet header. The minimum value of this field is 5 or 5 times 4 equals 20 bytes.
- Types Of Service: 8 bits field to use for differentiated services , such as prioritization of IP datagram.
- Total length: This is 16 bits field, and it gives the total length of the IP datagram.
- Identification: This field is of 16 bits and is used be the receiver to reassemble the packets.
- Flag fields have 3 bits: and one of them is reserved bit, other two are:
  - DF: Don't Fragment bit, when set to 1, the datagram cannot be fragmented.
  - MF: More Fragment bit, if this bit is set, it signifies more packets are

expected.

- Fragment Offset: This field specifies the position of data in this fragment in a complete message.
- Time To Live: TTL, its value signifies the maximum number of hops packets can travel before being discarded.
- protocol: Informs network layer of the protocol of the packet. for TCP it's 6.
- Header Checksum: Provides the checksum of IP Header and at every hop it gets verified, and if the checksum doesnt match datagram gets discarded.
- Source Address: 32 bits IP address of the device that sent the datagram.
- Destination Address: 32 bits IP address of the recipient of the datagram.
- Data: Data to be sent to a recipient. It can be fragmented or a complete message.

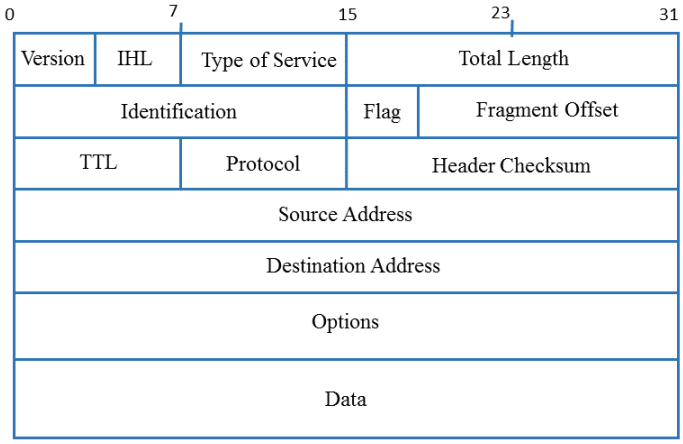

Figure 3-4: Diagram showinng fields of IP packet header.

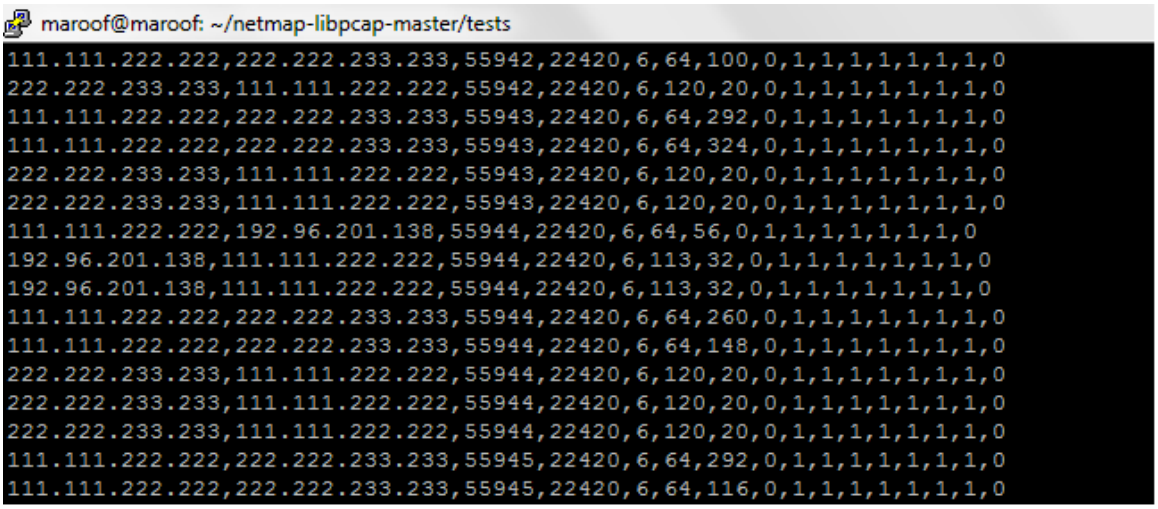

Figure 3-5: screenshot of csv files showing captured header fields.

In the figure 3-5, we have shown the screen shot of a file of captured packet headers. As we collect these data files we move them to HDFS for further processing of data. Every row in this field constitutes extracted header fields of one packet and every row in a file contain sixteen fields, separated by a comma. We have selected numerous header fields from TCP/IP traffic. These fields are taken into consideration after reviewing previous works for TCP-based intrusion detection. They are:

1. Source Packet.
2. Destination Packet.
3. Source Port.
4. Destination Port.
5. Protocol.
6. Time To Live.
7. Payload.
8. Window Size.
9. FIN Flag: Finished flag, no more data from a sender.
10. RST Flag: Reset the connection.

11. SYN Flag: Synchronize sequence numbers.
12. PSH Flag: Push Function.
13. ACK Flag: Acknowledgment field significant.
14. URG Flag: Urgent Pointer field significant.
15. CWR Flag: Congestion Window Reduced Flag.
16. ECN Flag: Explicit Congestion Notification Flag.

Once these, files are generated, we move these files onto HDFS every five minutes. These sixteen fields from the packet header helps us calculate the network packet features.

3.2 Packet Processor

In this section, we will describe the efficiency of the Spark framework for processing packet header in features calculation. To have a good measure of it, we analyzed the CAIDAs attack dataset consist of various attacks inside an organization [2]. The dataset comprises of the trace files of a DDoS attack which happened approximately for 40 minutes. Traces were split into pcap files for every five minutes with a cumulative size of 21.1 GB. Figure 3-6 presents the traffic volume for every five minutes. We observed that maximum traffic volume was 2.8 GB, in the interval of 30-35 minutes. We compared the processing time of this data by varying nodes in Spark cluster.

In the lab, for the system specification for our experimental setup, the server we have used is Supermicro SYS-6028RWTRT Intel Xeon (R) with 2.30 GHz, 20 CPU core 2.99 GHz, and 96 GB ram. The server also runs VMware ESXi host [20]. We have created virtual machines for monitored networks, Spark cluster & Hadoop file storage, and TCM using the ESXi host. The Spark cluster consists of seven nodes, out of which six worker nodes and one master node. For performance evaluation, we vary the number of workers from 1 node to 6 nodes. Each node in the Traffic cluster

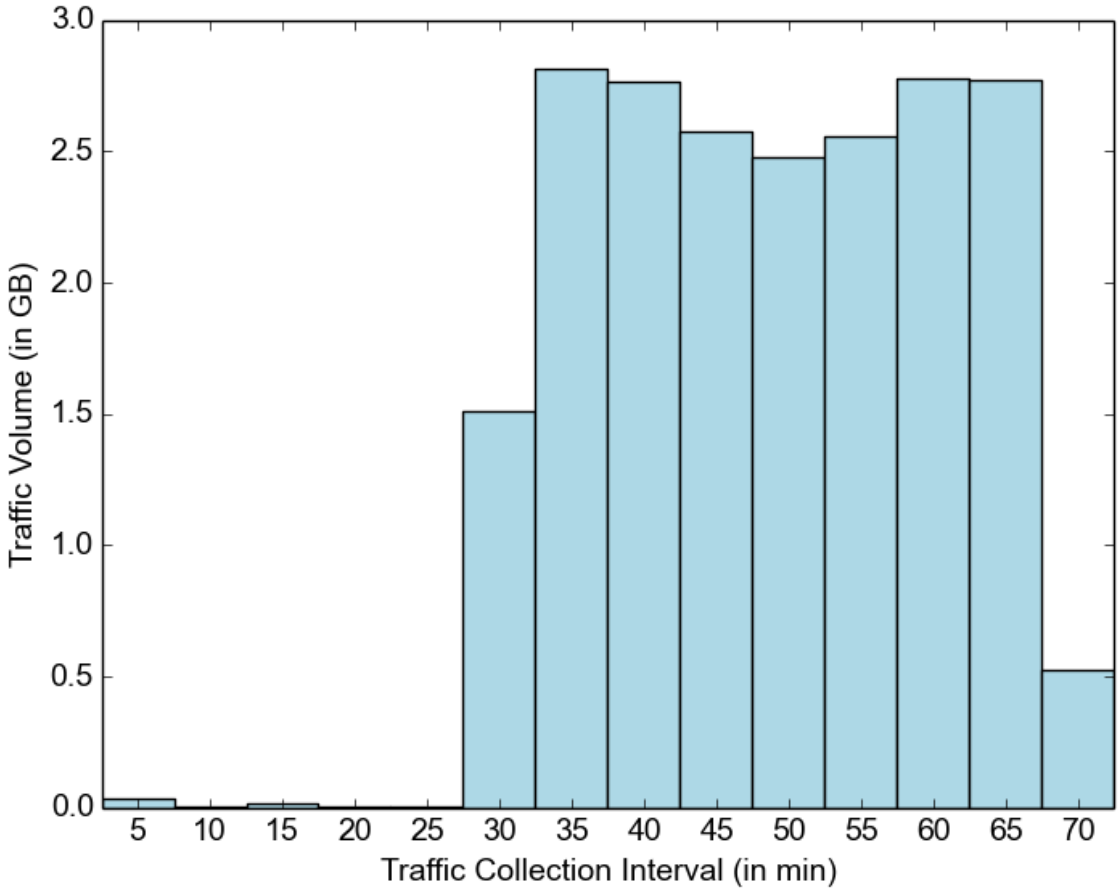

Figure 3-6: Traffic distribution during network intrusion, analyzed from CAIDA dataset [2].

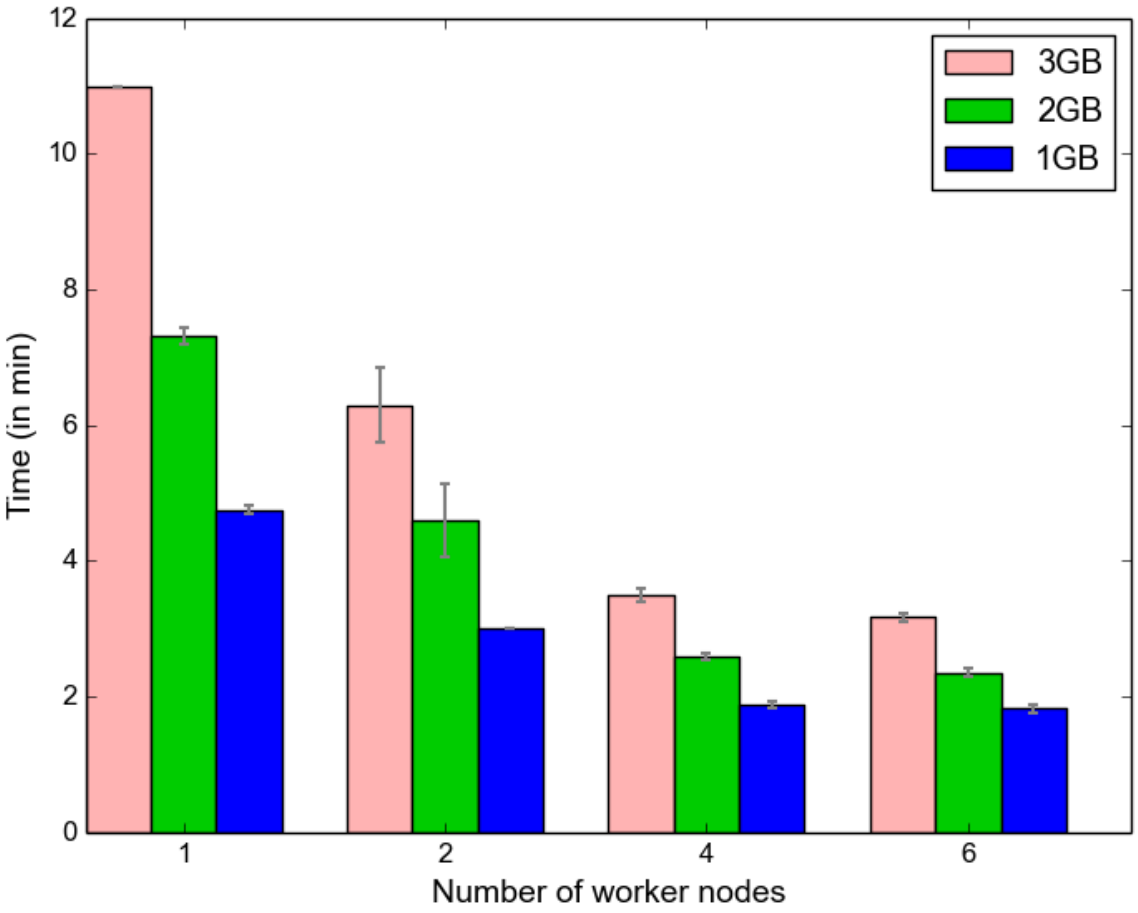

Figure 3-7: Time taken by varying file size to process with change in number of nodes.

and monitoring system and Spark cluster are assigned with 60 GB disk storage, 4 vCPUs, and 8 GB memory. We created 10 virtual machines with a configuration of 1 vCPU, 15 GB disk, and 2 GB RAM for the network to be monitored. In figure 3-7, we have shown the comparison between processing varying files of size 1GB, 2GB, and 3GB with different nodes of 6, 4, 2, and 1 machines. We were able to process the maximum traffic density of five minutes duration well under four minutes, giving us the near real-time traffic monitoring. The 3 GB files, created in 5 minutes, just took 3.5 ± 0.1 and 3.17 ± 0.05 minutes on 4 and 6 worker nodes, respectively. In figure 3-7 [20], we observe that, if the worker node is 1 or 2, it takes more than 5 minutes to process the 2 GB, and 3 GB files for feature extraction. However, it takes less than 5 minutes to process 1 GB of files in all cases. We can discern the pattern that as we increase the number of nodes, computing time decreases, and slowly the graph becomes steady indicating saturation. So, even if we increase the number of nodes, it will not affect the processing time. It is due to the overhead cost of moving data from one machine to other in the cluster.

Figure 3-8 [20] shows the screen shot of some of the features calculated from the header fields of the network packets. Based on the earlier works in the field of TCP/IP intrusion detection, we extracted the features in table 3.1 and given a brief description about them in the corresponding column. To make it clearer for our audience, we will explain following terms: 1) Flow: Each flow is defined by the combination of destination IP, source IP, destination port, source port, and protocol. 2) Symmetric Flow: Packet flow is considered symmetric only if source IP of the first packet in a pair of packet is same as the destination IP of the second packet in the same pair, source port of the first packet in the pair is same as the destination port of the second packet in the pair and protocol of both the packet in a pair should be same. 3) Asymmetric Flow: Those flows which do not fall in the above category is asymmetric flow.

Table 3.1: Traffic features extracted for TCP DDoS attack detection [20]

	Features	Description
1	IFF	Number of incoming total flows
2	OF	Number of outgoing total flows
3	FSF	Fraction of symmetric flows
4	FAF	Fraction of asymmetric flows
5	BPIF	Total number of bytes per incoming flows
6	BPOF	Total number of bytes per outgoing flows
7	PPIF	Number of packets for each incoming flows
8	PPOF	Number of packets for each outgoing flows
9	DSP	Total number of distinct source port for incoming flows
10	DDP	Total Number of distinct destination port for all incoming flows
11	FHDP	Fraction of destination port which are less than 1024
12	FLDP	Fraction of source port which are greater than 1024
13	FIPUSHS	Fraction of PUSH flag is set for incoming flows
14	FOPUSHS	Fraction of PUSH flag is set for outgoing flows
15	FISYNS	Fraction of SYN flag is set for incoming flows
16	FOSYNS	Fraction of SYN flag is set for outgoing flows
17	FIACKS	Fraction of ACK flag is set for incoming flows
18	FOACKS	Fraction of ACK flag is set for outgoing flows
19	FIFINS	Fraction of FIN flag is set for incoming flows
20	FOFINS	Fraction of FIN flag is set for outgoing flows
21	FIURGS	Fraction of URG flag is set for incoming flows
22	FOURGS	Fraction of URG flag is set for outgoing flows
23	FIRST	Fraction of RST flag is set for incoming flows
24	FORST	Fraction of RST flag is set for outgoing flows

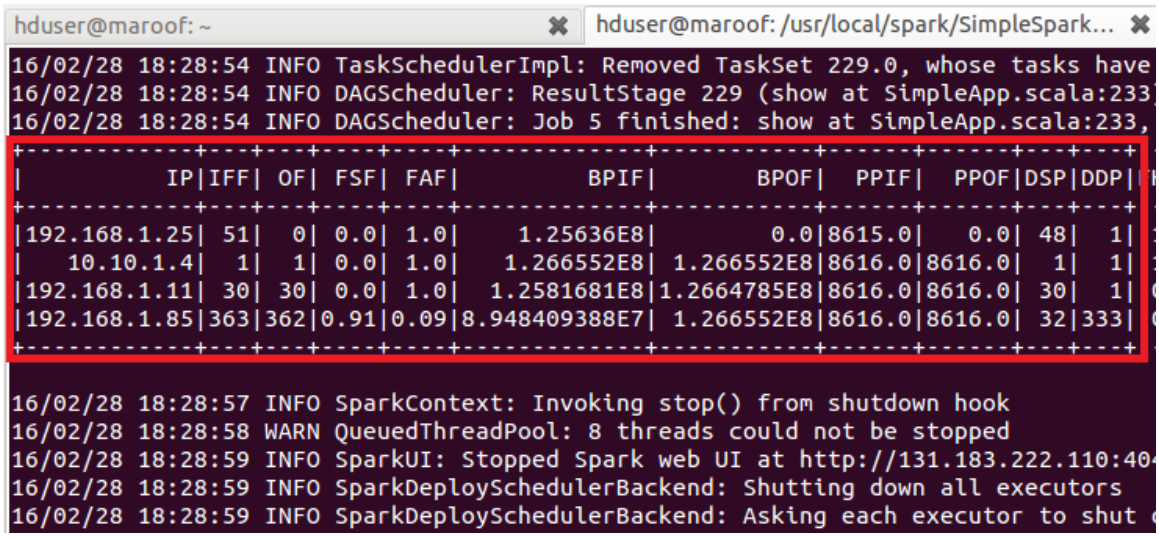

Figure 3-8: Snapshot of the features calculated by spark cluster.

Chapter 4

Traffic Classification

In chapter 3, we discussed the distributed IDS cluster we developed and its efficiency in monitoring TCP/IP attacks. In this chapter we will discuss the traffic datasets we have generated and collected for developing a machine learning model for IDS. After that, we will present, the various classifiers we have used and their results.

4.1 Traffic Dataset

In developing the classification model, we required internet traffic data of huge size to provide enough data to classifiers to train them properly. For this reason, we have collected data at different locations such as in our lab, department and home networks and also generated attack traffic in the isolated lab environment. To make it easier to label them, we split the collected dataset into two categories: Attack dataset and Normal traffic data.

4.1.1 Attack Dataset

As we have already mentioned, we have developed our application specifically for TCP/IP network attacks, so instead of using data available on the various websites which can have a mix of TCP traffic with network traffic of different protocols, we

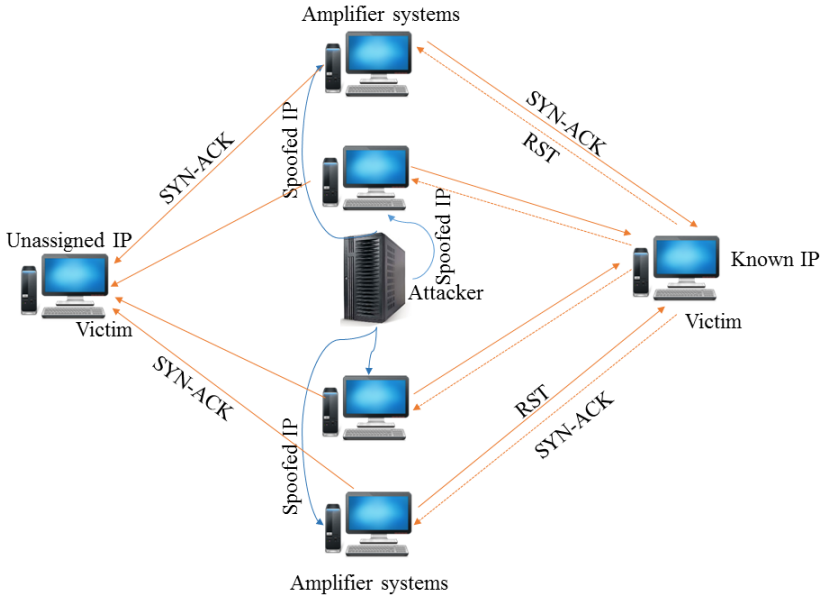

Figure 4-1: Figure showing the amplification attack setup.

decided to generate attack traffic only of TCP protocol. The attack tools we used are Hping3 and Sockstress. Here, we would like to give a brief explanation of some of the attacks and also describe the tools required to generate these attacks.

4.1.1.1 Types of TCP attacks

- Amplification attack: In an amplification attack, an attacker sends SYN request to the amplifier machines with the spoofed IP of a victim. Then the amplifier responds back with a packet with SYN-ACK flags set to the victim. If the packets are sent to the victim with assigned IP, it will reply with RST flag, and if the packets sent to the victim are unassigned IP, it will just populate the network. So, if multiple machines attack a single target, they can easily exhaust its resources and make the victim unavailable for legitimate traffic.

- SYN Attack: The foundation of SYN attack lies in the design principle of the three-way handshake. Every SYN request allocates a socket from the list

of limited available sockets for TCP connection. So, if the number of SYN connection request exhaust all the available sockets, the victim can no longer accept any new connection request. Before giving up on half-connection, it holds the connection momentarily and hence occupies a socket for that amount of time. So when the victim's network is flooded persistently with SYN request, this state is known as DoS, where actual SYN request denied setting up a connection. SYN attacks have multiple variants [29] and some of them are:

  - Direct Attack: In this attack category, attackers directly floods the victim with SYN request without the involvement of any third party. In this kind of attack, an attacker should make sure to maintain half-connection and shouldn't send ACK packet for acknowledging the connection.
  - Spoofed Attack: In spoofed attack, the victim is flooded by the SYN packets from an attacker using spoofed IP. Attacker spoof the source IP address of the packet such that source IP address does not respond to SYN-ACK packets from the victim, leaving the connection half open. This attack maintains the anonymity of the attacker.
  - Distributed Attack: The above two attacks involved just a single attacker and it is relatively easier to block them to prevent future attacks, however when the multiple machines are used by the attackers to attack a victim it becomes difficult to block, and the victim's network get flooded quickly. In these attacks, attackers use thousands of botnets available on the internet to attack victim by clogging its network.

- Sockstress [30]: To setup multiple TCP connections with the victim, Sockstress uses raw sockets. The main benefit is that connections are setup without saving any connection state on the attacker's machine because raw sockets are used. SYN cookies do not deter Sockstress attacks. It works such that in the last

packet of the three-way handshake, an attacker sends the packet with zero in window size. The zero window size indicates that the attacker is busy and is not able to accept more data. It causes the victim to keep the TCP connection alive, and it regularly probes the attacker to see if it can receive new packets. It leads the victim to exhaust its resources.

- TCP URG Flood: This attack floods a victim with TCP packets with URG flag set, from random and spoofed addresses. If the sent packets manage to guess parameters of existing connection, the victim will pass data immediately to an application layer for execution. However, it is highly unlikely, so these kinds of packets are meant to exhaust the resources of the network by prioritizing themselves.

- TCP RST and TCP FIN: These attacks are mostly intended to flood the victim's network. If they guess the parameters of connection correctly, they can close the established connection.

4.1.1.2 Attack Tools

All of the attacks described in section 4.1.1.1, can be generated mainly by two tools described below:

- Hping3 [18]: It is a tool for custom packet assembler and can be used for sending packets with the different protocols. We have used it to generate SYN, FIN, URG and RST flooding, by simple tweaking in the parameters. In the flood mode, it can send packets at the rate of over million packets per second. For Amplification attack, we spoofed the the source IP parameter of the packet before sending to amplifier machines.

- Sockstress [30]: To generate Sockstress attack, raw sockets are used and to generate raw sockets, the application should be run as root. We should update

IP tables so that OS shouldn't send RST packets to the victim. This causes the persistent connection during the attack. To execute an attack, it should not send RST packets to the victim in a case of any unrecognized SYN/ACK. Sockstress only needs three parameters to launch an attack, and these are victim IP, victim port, and a network interface to send packets like eth0.

4.1.1.3 Attack Testbed and data generation and collection

In this section, we will describe the testbed we build and the method used to generate the data. The testbed consists of 20 victim machines and ten attacker machines. Each attacker would flood the two victim machines with the frequency of the packets for each attack varying from 500 to 3000 packets per second with payload ranging between 100 to 500 bytes. Moreover, four out of these ten attackers would also target four victims with Sockstress attack, and other four victims will bear Amplifier attacks from all the attacker machines with the same frequency as mentioned above. In figure 4-2, we have shown a simplified diagram of the testbed, where machine A floods victim 1 & 2 and also machine A sends spoofed IP of victim 1 to generate amplification attack on 1. Same way machine B will flood victim 3 & 4 and it will send the spoofed IP of victim 3 to other attackers to generate amplification attack on 3 and so on for machine C, D and E. Machine F to J will flood two of the victims and also attack victims 11 to 15 with Sockstress attack. We continue to do this for 48 hours to collect enough data. Since we had just twenty victims, we used bittwiste [31] to edit half of the records to other 20 IPs so that we can double our data records. The total size of the traffic handled over two days was around 60 TB but before storing it we extracted only packet headers, and its total size was around 300 GB.

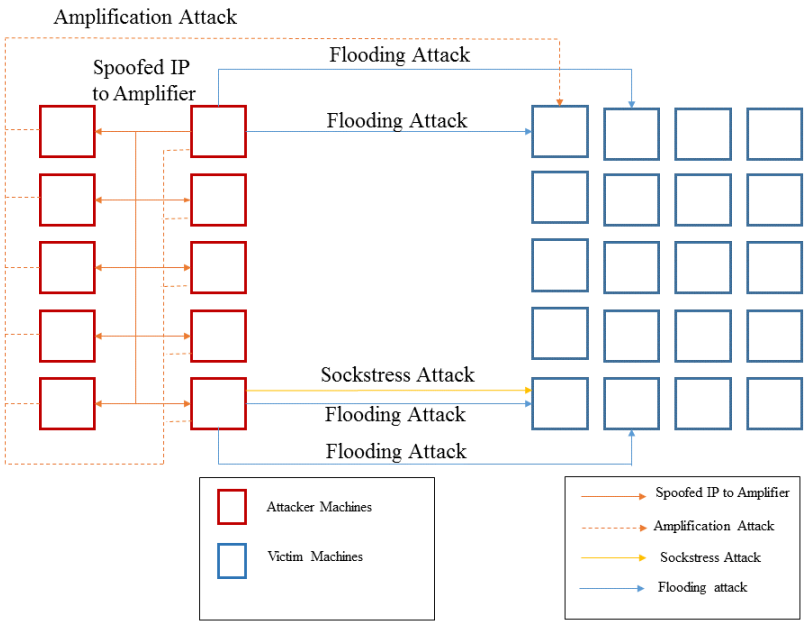

Figure 4-2: Figure showing the testbed setup to generate attack dataset.

4.1.2 Normal Dataset

For normal traffic, unlike attack dataset, we installed collector system in our department network and also in the home network to collect the traffic. The collector machine is a high-end server with a packet mirroring switch attached to it to redirect all the traffic passing through the network interface to it. We filtered out the TCP traffic for analysis. To accumulate an ample amount of data for machine learning purpose, we collected the traffic data for over seventy-two hours.

4.2 Traffic Dataset Analysis

In this section we will present the analysis done on the network traffic. First we will present the relationship between the two classes, i.e. anomalous and normal traffic and then the result from various machine learning classifiers. In the figure 4-3 and 4-4, we have shown the heat map of normal and anomalous traffic separately.

A heat map is a means of data visualization in the two-dimensional matrix where the individual values are represented as different shades of color. All the twenty-four features are on x-axis and data points on y-axis are plotted and features are numbered as in table 3.1. To make the visualization more clear, we have drawn the map on normalized dataset. Figure 4-5, shows the traffic pattern of merged dataset. The blue color show the values near zero and the red color show the values near one in a normalized data. If we compare the normal and anomalous traffic, we will notice that some of the features have similar values where as others have completely distinct values in both dataset.

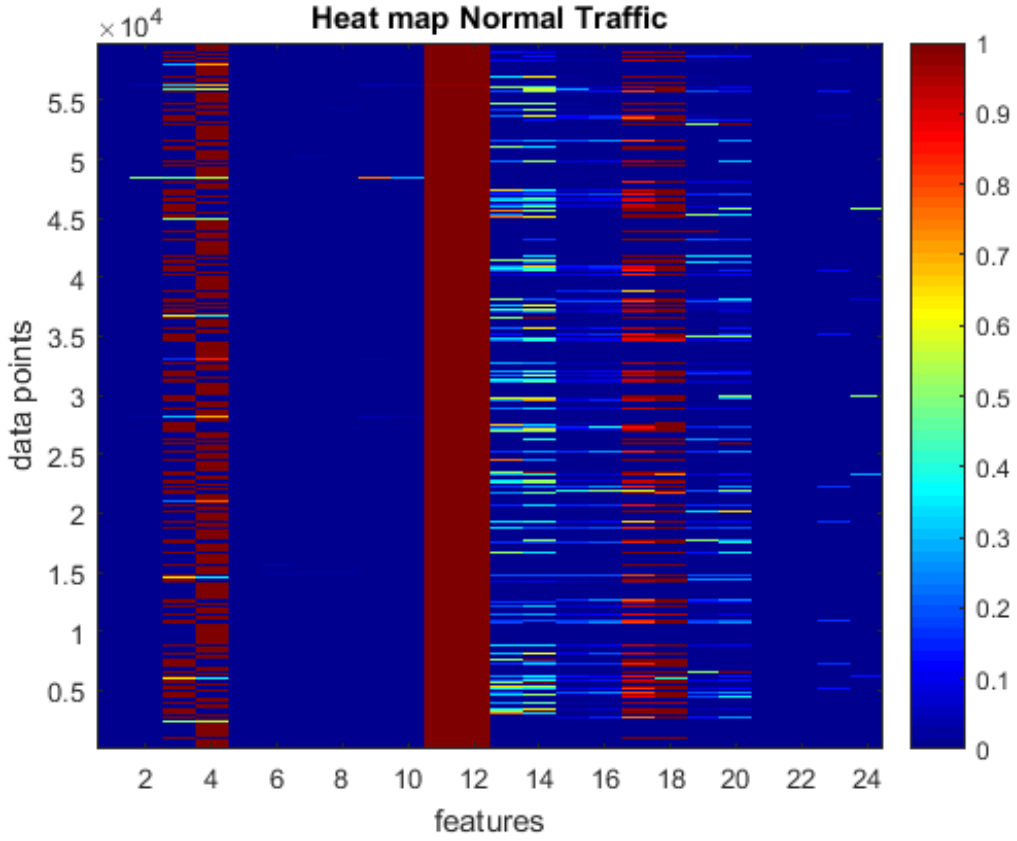

Figure 4-3: Figure showing the normal traffic pattern.

After analyzing the data, we used three machine learning algorithms to develop a

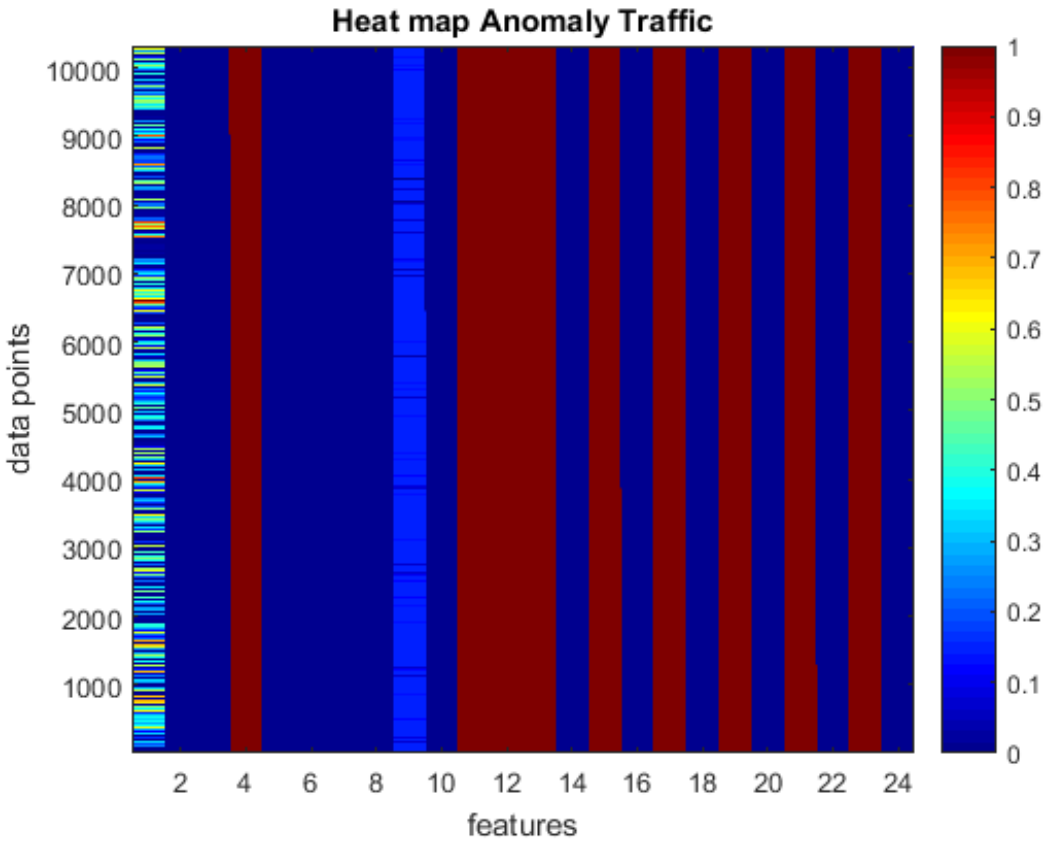

Figure 4-4: Figure showing the anomalous traffic pattern.

model to detect the future attacks. We have used the Spark-MLlib machine learning libraries for training and testing purpose. In the table 4.1, we have shown the output of naive-bayes model. To improve the accuracy further we used Support Vector Machine (SVM) algorithm. The figure 4-6, show the ROC curve generated by the output of the SVM model. The ROC or receiver operating characteristic curve is a method of visualizing the machine learning classifier performance [32]. In the figure 4-6, blue curve shows the roc curve of SVM with False Positive Rate on x-axis and True positive Rate on y-axis.

Where,

$$\text{True Positive Rate} = \frac{True\ Positive\ Values\ Indentified}{Total\ Positive\ Values} \tag{4.1}$$

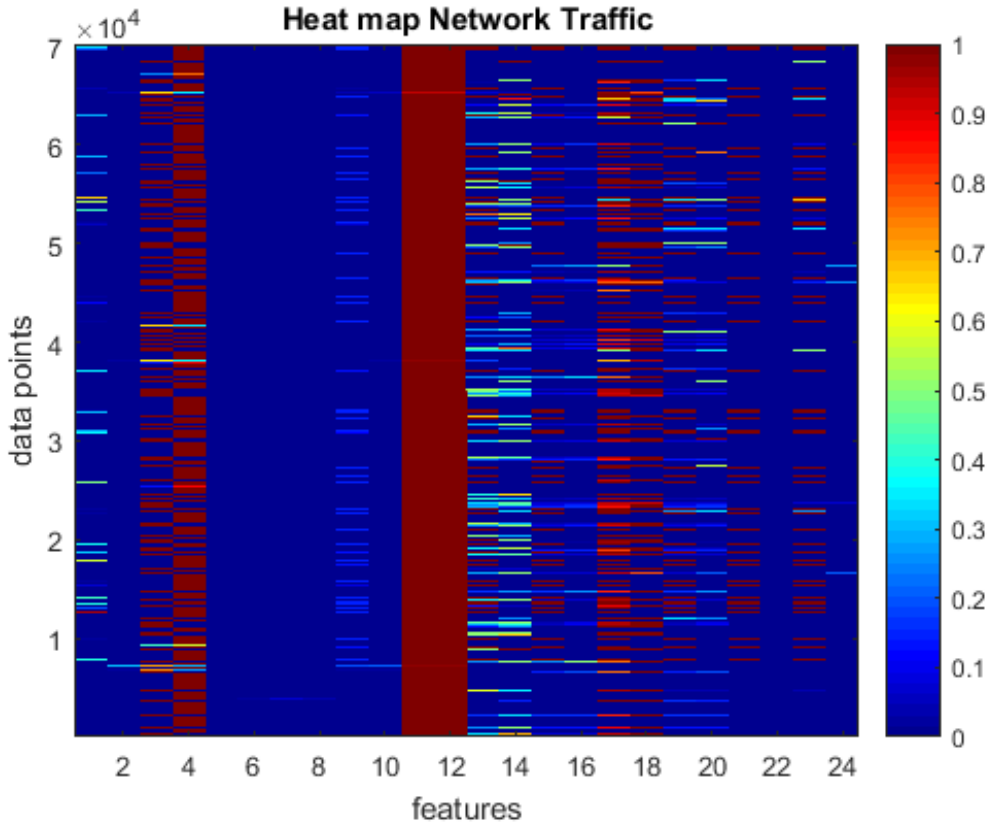

Figure 4-5: Figure showing the normal traffic pattern.

$$False\ Positive\ Rate = \frac{False\ Positive\ Values\ Indentified}{Total\ Negative\ Values} \tag{4.2}$$

The ROC curves gives us the idea about probability of making right guess for finding the true positive. The red line indicates that the probability of the classifier to classify true positive is 50% . The blue line of SVM classifier indicates it has high probability of finding the correct label. Area under the blue curve (AUC) is 0.93, which is good enough for practical applications.

The other classifiers we used are Decision Tree and random forest, we will focus on Random Forest as decision tree is a special case of random forest in which tree count is only one. The Random Forest classifier gives us 100% accuracy when the number of tree count in the algorithm is increased to ten. In figure 4-7, we have shown one

	Description	Value
1	Accuracy	98.8
2	Fraction of Attack packets detected correctly	0.93
3	Fraction of Normal packets detected correctly	0.99

Table 4.1: Analysis of network traffic using Naive-Bayes algorithm

of the tree from the random forest classifier. The main advantage of using random forest over other classifiers is, it avoids over fitting.

In the end, we will compare the accuracy, precision, recall and F-score of all three classifiers in figure 4-8 and 4-9. We will define these four terms in the following section with the help of the confusion matrix shown in table 4.2. Confusion matrix has four regions: 1) True Positive or TP is the number of times the prediction is positive for the positive label in the dataset. 2) True Negative or TN is the number of times prediction is negative for the negative label. 3) False Positive or FP is the number of times the prediction is positive for the negative label. 4) False Negative or FN is the number of times the prediction is negative for the positive label. Once we understand confusion matrix we can explain the following term [33] [34].

- Accuracy: It is the fraction of the sum of TP and FP by the sum of TP, TN, FP, and FN.

$$\mathrm{A}ccuracy = \frac{TP \ + \ FP}{TP \ + \ TN \ + \ FP + \ FN} \tag{4.3}$$

- Precision: It is the ratio of TP to the sum of TP and FP. Precision calculates the fraction of sample data predicted as positive that are truly positive. It is particularly useful when the dataset is highly skewed.

$$\mathrm{P}recision = \frac{TP}{TP \ + \ FP} \tag{4.4}$$

- Recall: It is the ratio of TP to the sum of TP and FN. Recall gives the measure

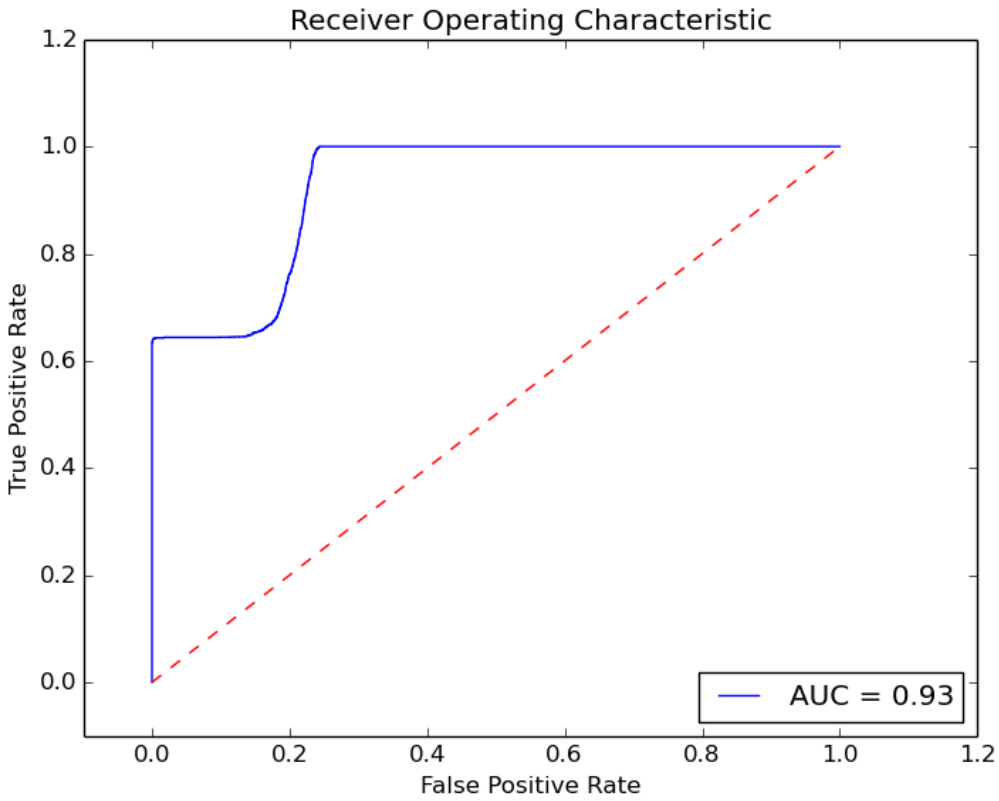

Figure 4-6: Receiver Operating Characteristic(ROC)from the SVM algorithm.

of the fraction of positive records correctly identified.

$$\text{R}ecall = \frac{TP}{TP \ + \ FN} \tag{4.5}$$

- F-score: The F-score is the harmonic mean of the values of precision and recall. It gives an idea about how accurate is the classifier and fraction of data points it is considering.

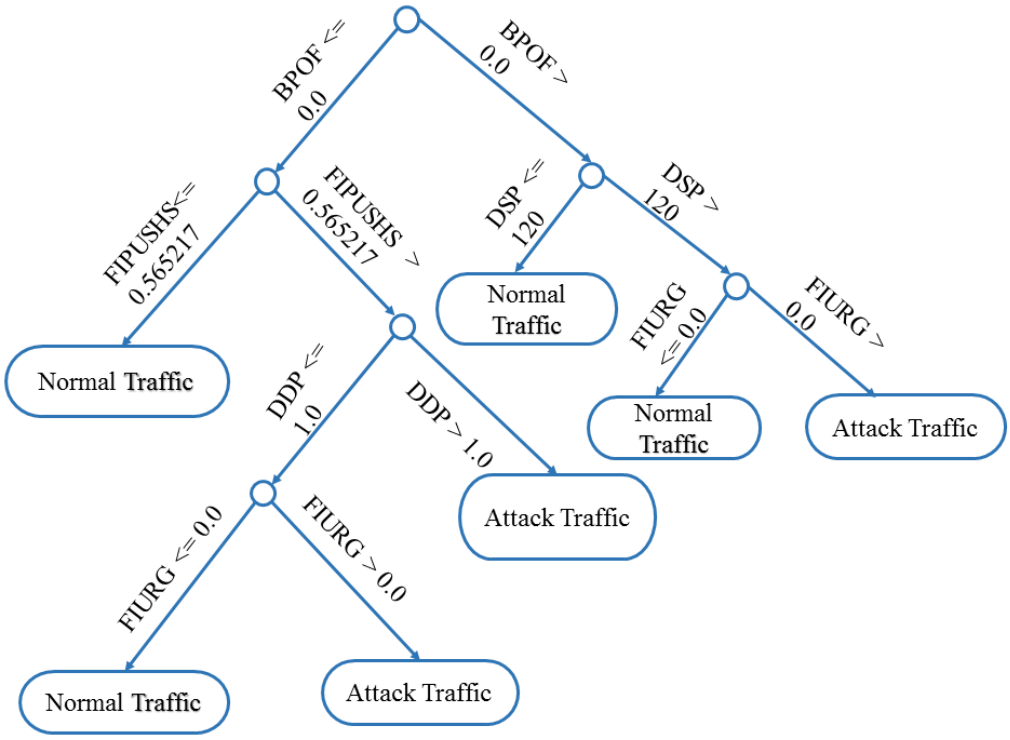

Figure 4-7: One of the decision tree from a random forest algorithm.

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	TN	FN

Table 4.2: Confusion Matrix

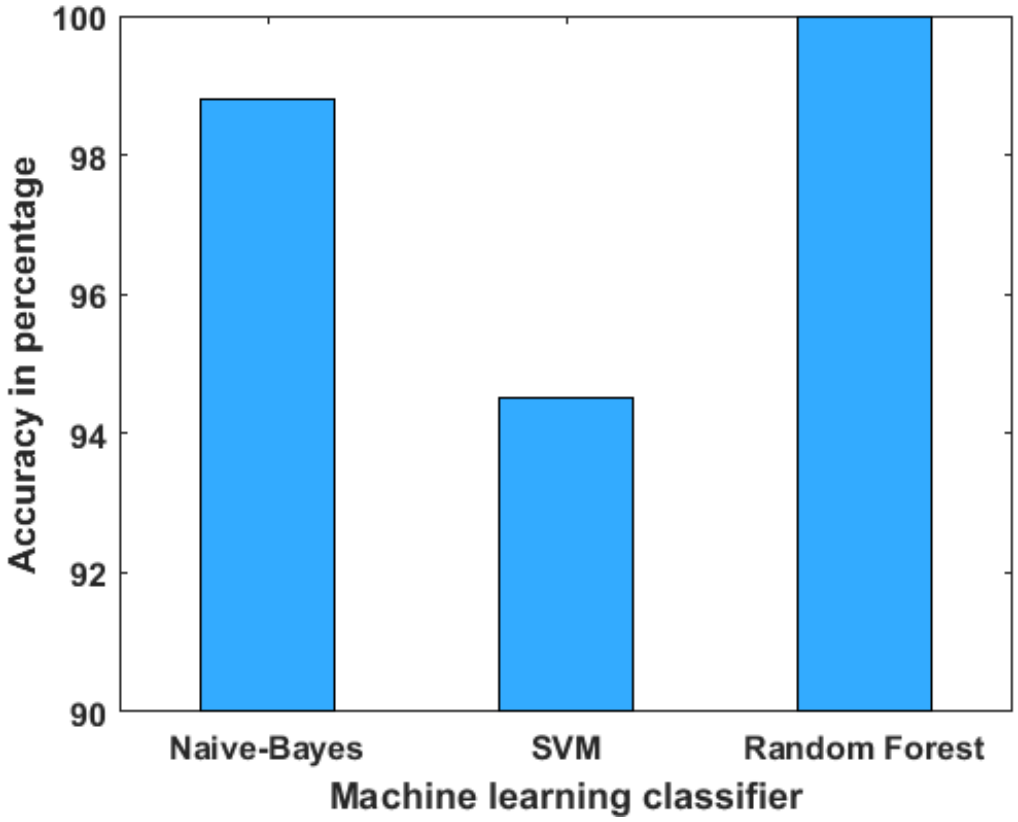

Figure 4-8: Graph showing the percentage accuracy of three classifier.

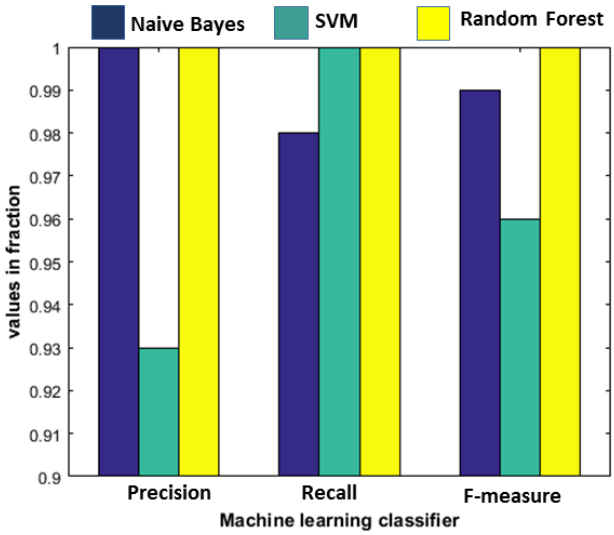

Figure 4-9: Graph comparing the precision, recall and F-measure of three classifier.

Chapter 5

Conclusion and Future Work

In this chapter we will summarize our work and then we will discuss some of the things we see can be used to enhance the IDS further.

5.1 Summary

The focus of our research work was to enhance the processing capacity of IDS and hence reduce the execution time to make the system near real-time. The other important part of our work was to build an IDS system such that it has the functionality to enhance the anomaly detection rule. To achieve this objective, we developed an application to use Spark-mllib machine learning classifiers.

In the chapter 3, we discussed the efficiency of spark distributed system we built for processing network packets for real-time analysis. We have shown in figure 3-7 the result of execution time, and it is evident from the bar graph that the time needed to process the network traffic decreases significantly on increasing the number of nodes. We were able to process CAIDA data quicker than the time taken to generate the same amount of data from the network.

In the chapter 4, we presented the detailed analysis of the traffic data collected and generated in the isolated lab environment. We presented the outcome of the various machine learning classifiers and depending on the result best one among them can

be chosen; In our case, the random forest classifier gave the perfect result. Spark machine learning libraries give us the advantage of using distributed resources and make the learning iterations notably fast. Moreover, just with little change, it's easy to switch between the different classifier and to compare the results among them.

We were able to achieve our objectives successfully but during our research, we realized that we could improve our work further. We will mention them in the next section.

5.2 Future Work

Some of the works that can be done to improve the application further are:

1. We have just a single switch to collect the traffic data; there's a possibility of it becoming a bottleneck at high-speed traffic. Instead of just one collector if we install the multiple collector points and merge them to HDFS, we can increase the data capturing capacity of the system.

2. We didn't explore Spark's streaming tool in this work; it can be applied to enhance the processing power of the IDS. Spark streaming can be more useful for real-time analytics.

3. We based our results on the attack traffic generated in the lab environment, so to make the results more generic, machine learning classifiers should be applied on actual attack traffic.

References

[1] Charles Kozierok. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, San Francisco, CA, USA, 2005.

[2] Center for Applied Internet Data Analysis. The CAIDA "DDoS Attack 2007" Dataset. `https://www.caida.org/data/passive/ddos-20070804_dataset.xml`, August 2007. Online; Last accessed :24-July-2016.

[3] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 20152020. `http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf`, 2016. Online; Last accessed :24-July-2016.

[4] Ahmad Sharifi, Akram Noorollahi, and Farnoosh Farokhmanesh. Intrusion detection and prevention systems (idps) and security issues. *International Journal of Computer Science and Network Security (IJCSNS)*, 14(11):80, 2014.

[5] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, 2000.

[6] Pedro García-Teodoro, JesúS Díaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.*, 28(1-2):18–28, February 2009.

[7] Yeonhee Lee and Youngseok Lee. Toward Scalable Internet Traffic Measurement and Analysis with Hadoop. *SIGCOMM Comput. Commun. Rev.*, 43(1):5–13, January 2012.

[8] Yeonhee Lee, Wonchul Kang, and Youngseok Lee. A hadoop-based packet trace processing tool. In *Proceedings of the Third International Conference on Traffic Monitoring and Analysis*, TMA'11, pages 51–63, Berlin, Heidelberg, 2011. Springer-Verlag.

[9] Kamaldeep Singha, Sharath Chandra Guntukub, Abhishek Thakura, and Chittaranjan Hotaa. Big data analytics framework for peer-to-peer botnet detection using random forests. volume 278, pages 488–497, September 2014.

[10] Jiangtao Luo, Yan Liang, Wei Gao, and Junchao Yang. Hadoop based Deep Packet Inspection System for Traffic Analysis of E-business Websites. In *Data Science and Advanced Analytics (DSAA), 2014 International Conference on*, pages 361–366, Oct 2014.

[11] Romain Fontugne, Johan Mazel, and Kensuke Fukuda. Hashdoop: A MapReduce Framework for Network Anomaly Detection. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 494–499, April 2014.

[12] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *ACM CoNEXT '10*, Philadelphia, PA, December 2010.

[13] Hajime Tazaki, Kazuya Okada, Yuji Sekiya, and Youki Kadobayashi. Matatabi: Multi-layer threat analysis platform with hadoop. In *3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, 2014.

[14] Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M. Munaf, P. D.

Torino, and Dario Rossi. Experiences of internet traffic monitoring with tstat. *IEEE Network*, 25(3):8–14, 2011.

[15] Hyunjoo Kim, Ikkyun Kim, and Tai-Myoung Chung. *Frontier and Innovation in Future Computing and Communications*. Springer, 2014.

[16] Arian Bar, Alessandro Finamore, Pedro Casas, Lukasz Golab, and Marco Mellia. Large-scale Network Traffic Monitoring with DBStream, a System for Rolling Big Data Analysis. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 165–170. IEEE, 2014.

[17] Luigi Rizzo. Netmap: A novel framework for fast packet i/o. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 101–112, Boston, MA, June 2012. USENIX Association.

[18] Sanfilippo, Salvatore and et al. . Hping. `http://www.hping.org/hping3.html`, June 2016. Online; Last accessed :24-July-2016.

[19] Apache Hadoop. The Apache Hadoop Softwre Foundation. `https://hadoop.apache.org/docs/r2.7.0/`, April 2015. Online; Last accessed :24-July-2016.

[20] Ahmad M Karimi, Quamar Niyaz, Weiqing Sun, Ahmad Y Javaid, and Vijay K Devabhaktuni. Distributed network traffic feature extraction for a real-time ids. 2016.

[21] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[22] Apache Spark. The Apache Sprk - Lightning-fast cluster computing. http://spark.apache.org/, June 2016. Online; Last accessed :24-July-2016.

[23] Apache Spark. Machine Learning Library (MLlib) Guide. http://spark.apache.org/docs/latest/mllib-guide.html, June 2016. Online; Last accessed :24-July-2016.

[24] Apache Spark. The Apache Spark - Naive Bayes - spark.mllib. http://spark.apache.org/docs/latest/mllib-naive-bayes.html, June 2016. Online; Last accessed :24-July-2016.

[25] Marti A. Hearst. Trends controversies: Support vector machines. *IEEE Intelligent System*, 13(4):18–28, 1998.

[26] Breiman, Leo and Cutler, Adele. Random Forest. https://www.stat.berkeley.edu/~breiman/RandomForests/, July 2016. Online; Last accessed :24-July-2016.

[27] Arbor Networks. Arbor Networks Attack Data. https://www.arbornetworks.com/arbor-networks-atlas-data-shows-the-average-ddos-attack-size-increasing, July 2015. Online; Last accessed :24-July-2016.

[28] Mark Sportack. TCP/IP First Step, December 2004.

[29] Wesley M. Eddy. Defenses Against TCP SYN Flooding Attacks. In *The Internet Protocol Journal - Volume 9, Number 4*, pages 2–16. CISCO, 2006.

[30] Jack C. Louis. Sockstress Tool. https://defuse.ca/sockstress.htm, February 2016. Online; Last accessed :24-July-2016.

[31] Addy Yeow Chin Heng. Libpcap-based Ethernet Packet Generator. http://bittwist.sourceforge.net/, April 2012. Online; Last accessed :24-July-2016.

[32] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[33] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[34] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July 1997.